

CAMBRIDGE

计算复杂性

Computational Complexity
A Conceptual Perspective

【以色列】 Oded Goldreich 著
张薇 韩益亮 杨晓元 译



国防工业出版社

National Defense Industry Press

计算复杂性

Computational Complexity
A Conceptual Perspective

[以色列] Oded Goldreich 著
张 薇 韩益亮 杨晓元 译

• 北京 •

本书由国家自然科学基金项目（61272492）资助翻译出版。
著作权合同登记 图字：军-2014-036 号

图书在版编目（CIP）数据

计算复杂性/（以）戈德里克（Goldreich, O.）著；张薇，韩益亮，杨晓元译.

—北京：国防工业出版社，2015.11

书名原文：Computational Complexity: A Conceptual Perspective

ISBN 978-7-118-10387-8

I. ①计… II. ①戈… ②张… ③韩… ④杨… III. ①计算复杂性-研究
IV. ①TP301.5

中国版本图书馆 CIP 数据核字（2015）第 251229 号

This is a Chinese edition of the following title published by Cambridge University Press:
COMPUTATIONAL COMPLEXITY ISBN 978-0-521-88473-0

© Oded Goldreich 2008

This Chinese edition for the People's Republic of China (excluding Hong Kong, Macau and Taiwan) is published by arrangement with the Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.

© Cambridge University Press and National Defense Industry Press 2015

This Chinese edition is authorized for sale in the People's Republic of China (excluding Hong Kong, Macau and Taiwan) only. Unauthorised export of this Chinese edition is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of Cambridge University Press and National Defense Industry Press.

本书简体中文版由 Cambridge University Press 授权独家出版发行。
版权所有，侵权必究。

※

国防工业出版社出版发行

（北京市海淀区紫竹院南路 23 号 邮政编码 100048）

涿中印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 31½ 字数 780 千字

2015 年 11 月第 1 版第 1 次印刷 印数 1—2000 册 定价 129.00 元

（本书如有印装错误，我社负责调换）

国防书店：（010）88540777

发行邮购：（010）88540776

发行传真：（010）88540755

发行业务：（010）88540717

原著者序

非常高兴为《计算复杂性》一书的中译本作序。我希望读者会喜欢本书的内容及观点。

在过去的 10 年中，我曾经三次访问中国，受到了清华大学理论计算机科学研究所的热情接待。我很高兴能多次与该研究所的学生交流，并被他们的学习能力和对计算复杂性及相关领域（如密码学）的兴趣所打动。

事实上，计算复杂性是计算理论的核心，而计算理论与程序设计理论则是计算机科学的数学基础。我相信，计算理论的概念框架和思想十分重要，并与计算机应用及许多学科有紧密联系。希望本书可以使中国读者对这些框架和思想有进一步的了解。

在这里我还要感谢清华大学的姚期智教授和复旦大学的赵运磊教授，感谢他们在我逗留中国期间的盛情款待。

Oded Goldreich

2015.9

在人类的长期实践经验中，由于资源（如时间）的匮乏，对效率的追求是一个传统而普遍性的做法。因此，什么样的计算任务可以被高效解决一直是人们考虑的核心问题之一。

为了系统化地解决上述问题，一个关键步骤是要对计算任务及解决任务的程序给出严格定义。20 世纪 30 年代出现的可计算理论（Computability Theory）提供了这些定义。可计算理论主要关注具体的计算任务，并研究解决这些任务的自动化程序（即计算设备及算法），它为我们研究算法所需的计算资源（如时间）打下了基础。当研究的重点转向解决某个特定任务（或某种特定类型的任务）的任意算法所需的计算资源时，就形成了计算复杂性理论（也称复杂性理论）研究。^①

复杂性理论是计算机科学基础理论的核心领域，它主要研究计算任务的固有复杂性（Intrinsic Complexity）。换句话说，典型的复杂性理论研究主要针对于解决某个（或某类）计算任务而非某个具体算法所需的计算资源。事实上，复杂性理论倾向于从计算资源出发并主要关注计算资源本身，研究对可以解决的计算任务类所需资源进行限制而造成的影响。因此，复杂性理论研究的是在有限时间（和/或其他有限的计算资源）内能完成何种任务。

半个世纪以来，复杂性理论形成了两个主要的研究方向。第一个方向是通过分析计算过程的演化，为计算问题的复杂性建立具体下界。因此，在某种意义上，这个方向的核心是对计算过程的“低层次（Low-level）”研究，电路复杂性及证明复杂性中的大部分研究属于这个分支。与此相反，第二个方向的研究目标是当无法精确地定义个别计算问题或概念时，在问题及概念间建立联系。这一方向可以看作是对计算的“高层次（High-level）”分析。NP 完全性理论、对近似性的研究、概率证明系统、伪随机性以及密码学都属于这个分支。

本书重点介绍第二个分支，原因如下：首先，这个研究分支中的许多已知结论显然在概念上非常重要，也就是说，这些已知结论提供了非常吸引人的概念，令该领域之外的人士也能理解。其次，这些概念解释起来并不需要过多的技术细节，因此，“高层次”的研究分支更适于本书讲述。^②

最后说明一下本书的性质。我们认为计算复杂性理论具有极其丰富的概念性内容，针对这些内容应该专门开设相关的讲座或课程。编写一部相应的教科书正是我们写作本书的动机和理念。

本书从概念的角度介绍复杂性理论，既可作为教科书，也可供自学使用。事实上，本书最初是针对想要学习复杂性理论的学生及将要从事复杂性理论教学的教师而写的，然而，我们希望本书对专业人士也能提供帮助，特别是当复杂性理论某个研究分支的专家想要了解其

① 与此相反，当研究的重点是设计并分析具体算法（而非计算任务的固有复杂性）时，则属于另一个称为算法设计与分析（Algorithmic Design and Analysis）的子领域。此外，算法设计与分析还可根据计算任务来自于数学、科学或工程领域而进一步细分。然而，复杂性理论在某种资源范围内可解决的任务间保持一致性（无论这些任务来自哪里）。

② 最后，选择这个研究方向最关键的原因在于，这是我们长期从事的专业领域。

他研究分支时。

我们还希望本书能引起读者对复杂性理论的兴趣，并使具备充分背景知识（能轻松理解抽象的讨论、定义及证明）的读者了解该领域。然而，真心期望大多数读者具备算法方面的基础知识，至少应熟知算法的概念。

本书讲述的重点是复杂性理论的几个子领域（见以下的内容组织及章节概要）。每一部分都从该领域中的一些直观问题开始，其中体现了相关概念。我们将讨论这些问题的重要性、对问题及概念定义方法的选择、问题的求解方法以及答案中体现的思想。我们认为，上述这些方面构成了每个子领域的核心内容。

我们注意到，在某些情况下，复杂性理论的基本概念有助于简化技术细节，事实上，本书中的许多结论其证明过程比标准的证明更简单而易于理解。

本书网址：<http://www.wisdom.weizmann.ac.il/~oded/cc-book.html>

内容组织及章节概要

本书包含 10 章和 7 个附录。各章节的内容构成了本书的核心，写作风格是教科书式的，而附录中提供了相关背景知识或其他观点，写作风格更像是文章综述。章节（及附录）的长度和顺序与其重要性无关，我们试图以最佳的逻辑顺序来安排章节顺序（即尽量不出现向后的引用）。

以下简要介绍各章及附录的主要内容，我们在 1.1.3 节中也对各章节作了介绍，而这里的介绍对于初学者来说更容易接受，且比每一章之前的介绍更简略。

第 1 章 引言及预备知识

引言部分是关于计算复杂性理论的一个高层次概述，并讨论了这个研究领域的某些特点。另外，引言中还包含了对本书研究方法及习惯表达方式的重要说明。预备知识部分主要讲述可计算理论的相关背景知识，可计算理论为计算复杂性理论研究提供了环境及平台。更重要的是，预备知识中还定义了搜索与判定问题、求解问题的算法及其复杂度等核心概念。另外，还给出了非一致计算模型（如布尔电路）的基本概念。

第 2 章 P、NP 和 NP 完全性

P-vs-NP 问题可以定性地描述为对问题求解的难度是否超过了判定给定解正确性的难度。另一种定义是寻找证明的难度是否大于验证其正确性的难度，即证明是否比验证更加困难。通常，人们认为对于以上两个等价问题的回答是：寻找（或证明）比判定（或验证）更困难，即 P 与 NP 是不同的。现在假设 $P \neq NP$ ，则面对 NP 类中的某个困难问题时，我们只能希望证明其不属于 P 类。这就是基于归约思想的 NP 完全性引起人们关注的原因。一般而言，称一个计算问题可以归约为另一个问题，如果给定求解后者中的一个（高效）算法，可以高效求解前者。如果 NP 类中任何问题都能归约为 NP 类中的某个问题，则称该问题是 NP-完全的。令人称奇的是，NP-完全问题确实存在，并且人们已经证明，数学及自然科学中的几百个计算问题都是 NP-完全问题。

第 3 章 P 与 NP 的变形

非一致的多项式时间（P/poly）是指由处理特定长度输入的设备所执行的高效计算。这里基本上忽略了构造此类设备的复杂性（即在一致性条件下），但更精细的定义（基于“接受建议的机器”）中允许对非一致性进行量化。这提供了 P 类的一种推广。相反，多项式层级（Polynomial-time Hierarchy, PH）则推广了 NP 类，其在描述 NP 类时使用了一类量化布尔公式，这种公式中存在量词和全称量词的转换次数是固定的。普遍认为每一次量词转换都为这种公式增加了额外的功能。通过证明 NP 包含于 P/Poly 中，可以在这两个不同类之间建立联系，进而证明了多项式时间层级结构将衰减到其第二层。

第 4 章 资源越多功能就越强大吗？

当使用“良好”的函数来确定算法所消耗资源时，更多的资源消耗当然可以完成更多的计算任务。然而，如果使用了“坏”的函数，资源的增加可能并不起什么作用。这里“良好”的函数是指计算这些函数不会超过其规定的资源消耗。由此可以得到一些具体断言，例如，存在一些问题可以在三次方时间内求解，但无法在平方时间内求解。在非一致计算模型下，不存在“良好”的函数问题，从而容易证明一些独立的结论。

第 5 章 空间复杂性

本章研究计算的空间复杂性，主要讨论两种极端情况。第一种情况是具有对数空间复杂度的算法，这类算法似乎是多项式时间算法的一个正常子集。第二种情况是多项式空间复杂度算法，这类算法几乎可以求解本书中所有的计算问题。本章提到的许多结论包括遍历（无向）图的对数空间算法，以及相关性强弱的有向图集到相关性较强的有向图集间的对数空间归约。这些结论有助于理解空间复杂性的基本性质，这些性质使其不同于时间复杂性。

第 6 章 随机性与计数

具有各种失效概率的概率多项式时间算法定义了复杂性类如 BPP 、 RP 和 ZPP 。本章的主要结构包括利用非一致的建议来模仿概率选择（即 $BPP \subset P/poly$ ），以及利用一个 $\exists\forall$ -量词序列来模仿双边概率错误（即 $BPP \subseteq \Sigma_2$ ）。关于计数问题（即对 NP -类问题解的数量计数），我们区分了精确计数与近似计数（在相对近似的意义上）。特别地， PH 中任意问题可以归约到精确计数类 $\#P$ ，而（对 $\#P$ 的）近似计数可以（概率地）归约到 NP 。其他相关内容包括 $\#P$ -完全性，寻找唯一解的复杂性，以及近似计数与生成几乎一致分布的解之间的关系。

第 7 章 困难性的用途

困难问题可以根据需要而“发挥作用”，其最著名的应用是在密码学中。这里的关键问题是要桥接“偶然”困难性（即最坏情况或平均情况的困难性）与“典型”困难性（即较强意义下的平均情况困难性）之间的缝隙。我们考虑两个与 $P \neq NP$ 有关的猜想。第一个猜想是存在一些指数时间内可解的问题，不能用较小（如多项式规模）的（非一致）电路类求解。我们将证明，这种最坏情况下的猜想可以转化为平均情况下的困难性结论，从而得到 BPP 类的非平凡去随机化（甚至于 $BPP = P$ ）。第二个猜想是 NP 中存在一些问题，易于生成其（已解的）实例，但对其他人来说很难求解。这一猜想可用单向函数的概念来理解，后者是指一些函数，易于求值但（在平均意义上）难求逆。我们证明，从一个在相对温和的平均意义上难求逆的函数出发，可以得到一类几乎处处难求逆的函数，后者又可以给出非常难求近似的谓词（称为困难核心谓词）。这些谓词可用于构造通用的伪随机数发生器，以及许多其他的密码学应用。

第 8 章 伪随机数发生器

计算理论从新的视角看待随机性：它认为一个分布是随机的（或伪随机的），如果不能利用高效算法区分其与均匀分布。这种模式最早将高效程序与多项式时间算法相关联，它也用

于这种区分程序的大量受限类中。伪随机数发生器的原型中涉及到高效的、可欺骗任意可行程序的发生器；也就是说，区分器可能是任意的概率多项式时间算法，并可能远比发生器自身更复杂。这些发生器被称为通用的，因为其输出可以安全地用于任意高效应用中。相反，为了去随机化，可以使用比区分器（表示要被去随机化的算法）更复杂的伪随机数发生器。利用这种方法，并根据各种困难性假设，可以相应地得到对 BPP 类的去随机化（包括完全去随机化，即 $BPP=P$ ）。其他形式的伪随机数发生器还包括能欺骗空间受限区分器的，甚至于更弱的发生器，只具有某种受限的随机性（如输出一对相互独立的序列）。

第 9 章 概率证明系统

随机性和交互式验证程序促成交互式证明系统的诞生，这类证明系统似乎比确定的证明更加强大。特别地， $PSACE \supseteq coNP$ 中的任意集合都存在交互式证明系统（如在不可满足的谓词公式集合中），而普遍认为 $coNP$ 中某些集合不存在 NP -证明系统。交互式证明包含在概念上有意义的零知识证明，后者是一类交互式证明，其中（验证者）除了相信断言正确之外，得不到任何信息。在合理的复杂性假设下， NP 中每个集合都有零知识证明系统。（这一结论在密码学中得到了大量应用。）第三种类型的概率证明系统体现于 PCP 模型中，代表概率可验证的证明。这些（冗余的） NP -证明在要检查的位置数量与对证明过程正确性的置信程度间取得折衷。特别地，在冗余的 NP -证明中读取常数数量的比特，可以得到较小的错误概率。 PCP 定理称 NP -证明可以高效地转化为 PCP 。对 PCP 的研究与对近似问题复杂性的研究紧密相关。

第 10 章 对复杂性要求的弱化

鉴于大量的计算问题显然不可解，很自然地想到这些问题的条件是否能弱化，使得问题对原始应用而言仍有意义，且存在可行的求解算法。本章利用近似和平均情况复杂性理论提出两种类型的弱化。近似的概念针对着计算问题自身，即对每个问题实例，扩展其允许的解集。在搜索问题中，这意味着求一些“足够接近”最优解的解，而在判定问题中，这意味着要区分“是”实例与远离“是”实例的实例。至于平均情况复杂度，注意：对这一概念的系统性研究要求开发一种非平凡的框架。该框架的一个主要方面是限制分布类型，使得一方面允许各种正常的分布，另一方面又能防止平均情况复杂性退化为最坏情况复杂性。

附录 A 复杂性类汇总

这部分包括了本书中提及的大部分复杂性类的自包含的定义。可分为两部分，分别指用算法及其资源（即图灵机的时间和空间复杂度）定义和用非一致性的电路（及电路规模和深度）来定义的复杂性类。具体地，定义了以下复杂性类： P 、 NP 、 $coNP$ 、 BPP 、 RP 、 $coRP$ 、 ZPP 、 $\#P$ 、 PH 、 E 、 EXP 、 $NEXP$ 、 L 、 NL 、 RL 、 $PSPACE$ 、 $P/poly$ 、 NC^k 和 AC^k 。

附录 B 寻求下限

本附录简要综述对计算问题复杂性求下限的最著名的尝试。第一部分是电路复杂性，描述了解自然计算问题的（受限）电路的规模下限。这代表了一个计划，其长远目标是证明 $P \neq NP$ 。第二部分主要围绕证明的复杂性，描述对自然重言式的（受限制的）命题证明的长度下界。这个计划的长远目标是证明 $NP \neq coNP$ 。

附录 C 现代密码学基础

这是关于密码学基础理论的综述，重点讨论用于为自然的安全问题提供概念、定义及解决方案的范式、方法和技术。其中提出了一些概念性工具，以及利用它们得到的一些基本结论。附录中还加入了对单向函数、伪随机数发生器、零知识证明（见第 7 章～第 9 章）的讨论。利用这些基本工具，本附录描述了基本的密码学应用，如加密、签名及通用的密码协议。

附录 D 概率论基础及随机性中的前沿问题

概率论基础部分包括随机变量的相关知识和 3 个有用的不等式（即马尔可夫不等式、切比雪夫不等式和切诺夫界）。前沿问题包括 Hash 函数类的构造及引理、对采样的研究、估算任意一个函数平均值的随机性复杂度，和随机提取问题（即从弱的或有缺陷的随机源中提取几乎完美的随机数）。

附录 E 明确的构造

复杂性理论清晰地解释了明确构造的直观概念。本附录中用纠错码和扩张图来阐述复杂性理论观点。从纠错码开始，主要讨论纠错码计算的各个方面，回顾了几种流行的构造，并构造了具有常数码率和常数相对距离的二元码。本附录还简要回顾了本地可测试及本地可译码的概念，并对接近于任意单个序列的码字数量给出了一个有用上界。至于扩张图，附录中回顾了扩张图的两种标准定义，两种明确性水平，与（单步和多步）随机游走相关的扩张图的两个性质，以及扩张图的两种明确的构造。

附录 F 一些省略的证明

本附录包含正文中省略（由于种种原因）的部分证明，了解这些证明仍是有益的。其中包括利用随机 Karp-归约证明 $\mathcal{PH}$ 可归约为 $\#P$ ，以及关于交互式证明系统的两种有用变换。

附录 G 一些计算问题

本附录给出正文中所涉及的大部分具体计算问题的定义。特别地，还包括了图算法、布尔公式和有限域的简介。

我对于复杂性理论的观点在很大程度上受 Shimon Even 和 Leonid Levin 的影响。事实上，很难不受这两个杰出的、富于思想的研究者影响（特别是像我这样有幸与他们共事很长时间的人）。^①

Shimon Even 认为，复杂性理论研究的是算法的局限性，它与自然的计算资源及自然的计算任务相关。因而，复杂性理论可以指导工程技术人员，还可帮助求知若渴的计算机科学家们处理最深层次的问题。我相信本书分享了 Shimon 的观点，因为它正是围绕着上述问题而写成的。

Leonid Levin 强调复杂性理论的一般原则，反对任何“依赖于模型的效应”以及将复杂性理论与形式语言和自动机理论相提并论的传统做法。我认为，本书受 Leonid 的影响极深。

感谢许多同事在专业观点上的帮助，包括 Shafi Goldwasser、Dick Karp、Silvio Micali 和 Avi Wigderson。当然，这里只列出了一部分对我有帮助的人。

我在哈佛大学拉德克利夫研究院（Radcliffe Institute）进修期间，决定写作这本书。感谢拉德克利夫为我创造了一个宽松的环境。还要感谢许多同事针对本书早期版本提出的建议（或帮助），包括（不完全）Noga Alon、Noam Livne、Dieter van Melkebeek、Omer Reingold、Dana Ron、Ronen Shaltiel、Amir Shpilka、Madhu Sudan、Salil Vadhan 和 Avi Wigderson。

最后，非常感谢 Mohammad Mahmoody Ghidary 和 Or Meir 仔细阅读了本书初稿，感谢他们大量的校对工作及提出的宝贵建议。

与已有工作的关系

本书中某些内容摘自我以前做的一些工作。特别地，第 8 章和第 9 章分别基于我的一篇综述^[90]中的第 3 章和第 2 章而写成。但为了符合本书的写作目的而对阐述方法做了较大修改。类似地，7.1 节及附录 C 基于参考文献[90]的第 1 章和参考文献[91, 92]而写成，但是与这些工作有明显不同。相反，附录 B 摘自 Avi Wigderson 和我自己的文章^[107]，并修改得相对较少。

^① Shimon Even 是我的硕士导师（以色列理工学院，Technion，1980—1983），而我在做博士后工作期间（麻省理工学院，MIT，1983—1986）与 Leonid Levin 接触较多。

第 1 章 引言及预备知识

1.1	引言	1
1.1.1	复杂性理论概述	2
1.1.2	复杂性理论的特征	5
1.1.3	本书内容概要	6
1.1.4	写作方法与风格	9
1.1.5	标准符号及习惯性用法	12
1.2	计算任务及模型	13
1.2.1	表达方式	14
1.2.2	计算任务	15
1.2.3	一致性模型 (算法)	16
1.2.4	非一致性计算模型 (电路及建议)	28
1.2.5	复杂性类	33
	本章注释	33

第 2 章 P、NP 和 NP-完全性

2.1	P-vs-NP 问题	36
2.1.1	搜索版本: 求解与检验	37
2.1.2	判定版本: 证明与验证	39
2.1.3	两种表示的等价性	42
2.1.4	对 NP 的两个技术性说明	43
2.1.5	NP 的传统定义	43
2.1.6	对 P 不同于 NP 的支持	44
2.1.7	哲学思考	45
2.2	多项式时间归约	46
2.2.1	归约的一般概念	46
2.2.2	优化问题到搜索问题的归约	48
2.2.3	搜索问题的自归约性	50
2.2.4	总结及一般性观点	52
2.3	NP-完全性	53
2.3.1	定义	53

2.3.2	NP-完全问题的存在性	54
2.3.3	一些常见的 NP-完全问题	56
2.3.4	既不属于 P 也非 NP-完全的 NP 集	65
2.3.5	对完全问题的思考	67
2.4	三个前沿性问题	69
2.4.1	承诺问题	69
2.4.2	NP 问题的最优搜索算法	73
2.4.3	coNP 类及其与 NP 的交集	74
	本章注释	77
	习题	78

第 3 章 P 与 NP 的变形

3.1	非一致的多项式时间	86
3.1.1	布尔电路	87
3.1.2	接受建议的机器	88
3.2	多项式时间层级	90
3.2.1	量词的转换	91
3.2.2	非确定型预言机	93
3.2.3	P/poly-vs-NP 问题及 PH 类	95
	本章注释	97
	习题	97

第 4 章 资源越多功能就越强大吗?

4.1	非一致的复杂性层级	103
4.2	时间层级及缝隙	104
4.2.1	时间层级	104
4.2.2	时间缝隙及加速	109
4.3	空间层级和缝隙	112
	本章注释	112
	习题	113

第 5 章 空间复杂性

5.1	预备知识及相关问题	116
5.1.1	几个重要的习惯性表达	116
5.1.2	有用的最少计算空间	117
5.1.3	时间与空间	117
5.1.4	电路求值	123

5.2 对数空间	123
5.2.1 L 类	123
5.2.2 对数空间归约	123
5.2.3 对数空间一致性及更强的概念	124
5.2.4 无向连通性	125
5.3 非确定的空间复杂性	130
5.3.1 两个模型	130
5.3.2 NL 及有向连通性	131
5.3.3 回顾与讨论	137
5.4 PSPACE 及游戏	137
本章注释	140
习题	140

第 6 章 随机性与计数

6.1 概率多项式时间	149
6.1.1 基本模型	149
6.1.2 双边错误: BPP 类	152
6.1.3 单边错误: RP 和 coRP 类	155
6.1.4 零边错误: ZPP 类	159
6.1.5 随机的对数空间	160
6.2 计数	162
6.2.1 精确计数	162
6.2.2 近似计数	169
6.2.3 对唯一解的搜索	173
6.2.4 解的均匀生成	176
本章注释	182
随机算法	183
计数问题	184
习题	185

第 7 章 困难性的用途

7.1 单向函数	195
7.1.1 困难实例的生成与单向函数	195
7.1.2 弱单向函数的放大	198
7.1.3 困难核心谓词	201
7.1.4 对困难放大的思考	206
7.2 E 中的困难问题	206
7.2.1 对多项式规模电路的放大	208

7.2.2 对指数规模电路的放大	219
本章注释	225
习题	226

第 8 章 伪随机数发生器

8.1 通用模式	235
8.2 通用的伪随机数发生器	236
8.2.1 基本概念	237
8.2.2 典型应用	237
8.2.3 计算不可区分性	240
8.2.4 扩张函数的放大	243
8.2.5 构造	245
8.2.6 非一致的强伪随机数发生器	247
8.2.7 更强的概念及思考	249
8.3 对时间复杂性类的去随机化	250
8.3.1 正则去随机性发生器	250
8.3.2 正则去随机性发生器的构造	253
8.3.3 技术上的变化及概念思考	256
8.4 空间受限的区分器	257
8.4.1 定义	258
8.4.2 两种构造	260
8.5 特殊用途的伪随机数发生器	265
8.5.1 两两独立发生器	266
8.5.2 小偏移发生器	269
8.5.3 扩张图上的随机漫游	272
本章注释	273
习题	276

第 9 章 概率证明系统

9.1 交互式证明系统	288
9.1.1 动机和观点	288
9.1.2 定义	290
9.1.3 交互式证明的功能	292
9.1.4 变形及更好的结构: 概述	297
9.1.5 计算能力受限的证明者: 概述	299
9.2 零知识证明系统	300
9.2.1 定义	301
9.2.2 零知识证明的功能	303

9.2.3 知识的证明——附加内容	308
9.3 概率可检验证明系统	309
9.3.1 定义	310
9.3.2 概率可检验证明的功能	311
9.3.3 PCP 与近似	325
9.3.4 对 PCP 自身的更多讨论: 概述	326
本章注释	329
习题	331

第 10 章 对复杂性要求的弱化

10.1 近似	339
10.1.1 搜索或优化	340
10.1.2 判定或属性检测	344
10.2 平均情况复杂性	348
10.2.1 基础理论	349
10.2.2 研究分支	359
本章注释	367
习题	368

附录 A 复杂性类汇总

A.1 预备知识	375
A.2 基于算法的类	376
A.2.1 时间复杂性类	376
A.2.2 空间复杂性类	378
A.3 基于电路的类	379

附录 B 寻求下限

B.1 预备知识	380
B.2 布尔电路的复杂性	381
B.2.1 基本结论和问题	382
B.2.2 单调电路	383
B.2.3 有界深度电路	383
B.2.4 公式规模	384
B.3 算术电路	384
B.3.1 单变量多项式	385
B.3.2 多变量多项式	385
B.4 证明的复杂性	386

B.4.1	逻辑证明系统	388
B.4.2	代数证明系统	388
B.4.3	几何证明系统	389

附录 C 现代密码学基础

C.1	引言与预备知识	390
C.1.1	基本原则	390
C.1.2	计算模型	392
C.1.3	内容组织	393
C.2	计算困难性	393
C.2.1	单向函数	394
C.2.2	困难核心谓词	395
C.3	伪随机性	395
C.3.1	计算不可区分性	396
C.3.2	伪随机数发生器	396
C.3.3	伪随机函数	397
C.4	零知识	398
C.4.1	仿真范式	398
C.4.2	确切定义	399
C.4.3	一般性结论及应用	400
C.4.4	其他定义及相关概念	401
C.5	加密方案	403
C.5.1	定义	404
C.5.2	构造方法	406
C.5.3	超越窃听的安全性	407
C.6	签名和消息认证	407
C.6.1	定义	408
C.6.2	构造方法	409
C.7	通用的密码协议	411
C.7.1	定义方法及模型	411
C.7.2	一些已知结论	414
C.7.3	构造方法及两个简单协议	415
C.7.4	结语	418

附录 D 概率论基础及随机性中的前沿问题

D.1	概率论基础	420
D.1.1	符号约定	420
D.1.2	3个不等式	421
D.2	散列	424

D.2.1	定义	424
D.2.2	构造	425
D.2.3	剩余 Hash 引理	425
D.3	采样	428
D.3.1	定义的环境	429
D.3.2	已知结论	429
D.3.3	碰撞器	431
D.4	随机提取器	431
D.4.1	定义及各种观点	432
D.4.2	构造	436

附录 E 明确的构造

E.1	纠错码	439
E.1.1	基本概念	439
E.1.2	几种流行的码	441
E.1.3	两个计算问题	444
E.1.4	列表译码的上界	445
E.2	扩张图	446
E.2.1	定义和性质	446
E.2.2	构造	451

附录 F 一些省略的证明

F.1	证明 $\mathcal{PH}$ 可归约为 $\#P$	455
F.2	证明 $IP(f) \subseteq \mathcal{AM}(O(f)) \subseteq \mathcal{AM}(f)$	460
F.2.1	利用 $\mathcal{AM}$ -游戏模拟通用的交互式证明	460
F.2.2	$\mathcal{AM}$ 的线性加速	465

附录 G 一些计算问题

G.1	图	470
G.2	布尔公式	472
G.3	有限域、多项式及向量空间	473
G.4	矩阵的行列式与常值	473
G.5	素数与合数	474
	参考文献	475
	后记	485

第 1 章

引言及预备知识

当您前往伊萨卡时，应祈愿道路要漫长，
充满奇遇，充满新知。
——K.P.Cavafy，“伊萨卡”

本章包含两部分。第一部分对（计算）复杂性理论进行一个高屋建瓴的介绍，比前言中的简述更加详细，但是考虑到该领域内容的丰富性，本章的介绍也只能算是窥豹一斑。此外，在引言部分还就本书的内容、方法及写作风格给出了重要注解。



本章第二部分提供了阅读本书其余章节所必备的基础知识，包括计算任务和计算模型，以及依赖于计算模型的复杂性测度。具体地，这一部分将介绍计算理论中的基本概念和基本结论（包括图灵机（Turing Machines）的定义、关于不可判定性的结论、通用机（Universal Machines）及预言机（Oracle Machines）的概念）。另外，还介绍了非一致性计算模型的基本概念（如布尔电路（Boolean Circuits））。

1.1 引言

引言包含两部分内容：一是复杂性理论本身；二是对本书的一些介绍。第一部分是复杂性理论内容梗概（1.1.1 节），以及对其特征的思考（1.1.2 节）。第二部分叙述本书各章的内容（1.1.3 节）、内容取舍和表述方式（1.1.4 节），以及各种定义及习惯性表达（1.1.5 节）。

1.1.1 复杂性理论概述

辛苦的工作是甜蜜的^①。

Judges, 14:14

复杂性理论研究计算任务的固有复杂性，其“终极”目标是确定任意有明确定义的计算任务的复杂度。其他目标还包括解释各种计算现象之间的关系（如在与计算复杂性相关的各种事实之间建立联系）。事实上，可以认为前一个目标考虑的是特定计算现象的“绝对”解，而后一个则考虑计算现象之间的关系。

有趣的是，迄今为止，复杂性理论在达成后一个（“相对的”）目标上更为成功。实际上，正是由于在解决“绝对”问题时遇到的挫折，引发了人们解决“相对”问题的积极性。仔细想来，在一般情况下，如果很难找到绝对解，人们就会很自然地转而寻求有条件的解，由此可能发现计算现象之间的有趣联系。此外，无法彻底理解个别计算现象的情形似乎有助于开发将不同计算现象相关联的方法。总之，这就是复杂性理论的研究现状。

让我们把由于无法获得绝对解而产生的沮丧先抛开不谈，必须承认的是，在不同问题间成功建立联系的想法也很诱人：在某种意义上，计算现象之间的关系比个别现象的绝对解更具启发性。此刻，我们脑海中出现的第一个例子就是 NP-完全性理论。下面将从复杂性理论两个目标的角度引入该理论。

复杂性理论无法判定某些任务的固有复杂性，如求给定（可满足）命题公式的可满足性赋值，或求给定（3-着色）图的 3-着色方案。然而，复杂性理论证明，这两个看似无关的计算任务在某种意义下是一回事（或更准确地，它们在计算上等价）。这个结论出乎意料且激动人心，希望读者有同感。复杂性理论的其他许多发现也同样令人赞叹。事实上，读者将在本书中快速浏览构成复杂性理论的其他问题及结论。

我们从“P 与 NP 问题”（以下记作 P-vs-NP，译者注）说起。根据日常经验，通常人们认为，解决一个问题要比检验给定解的正确性更加困难（例如，迷宫或其他一些搜索问题）。这种经验仅仅是巧合，还是代表了一个基本事实（或世界的性质）呢？我们能否想象一个世界，在其中解任何问题并不比检验解的正确性更加困难？术语“解一个问题”在这个假想的世界中会不会失去其含义呢（这在我们的惯常思维中是不可能的）？否定这样一个假想世界的存在（即在其中“解决问题”不比“验证解”更困难），就是“P 不同于 NP”的实际含义。其中 P 表示可以有效解决的计算任务，而 NP 表示可以其解的正确性可以被有效检验的计算任务。

偏爱数学（或理论）的读者也许还会联想到证明一个定理与验证证明的正确性这两个任务。事实上，寻找证明正是上述“解一个问题”任务的特殊类型（而验证证明过程的正确性，则是验证解的正确性的相应情形）。在这里，“P 不同于 NP”意味着存在一些定理，证明它们要比验证给定证明的正确性更困难。这就意味着“证明”的概念是有意义的，即证明确实有助于确定断言的正确性。这里 NP 代表那些在提供了充分证明时可以被高效验证的断言类，而 P 代表在无证明时可以被高效验证的断言类。

根据前面的讨论，显然，P-vs-NP 问题是一个意义深远的基本科学问题。正是由于这一问题超出了我们目前的解决能力，才促进了 NP-完全性理论的发展。不严格地说，该理论确

^① 通常引用这句话来表达“祸兮福之所倚”。

定了一系列与 NP 问题难度相当的计算问题。换言之，P-vs-NP 问题的命运取决于这些问题中的任意一个：如果这些问题中的任意一个是易解的，则 NP 类中所有的问题都是易解的。因此，证明一个问题的 NP-完全性就提供了其不可解的证据（当然，这是在“P 不同于 NP”的假设下）。事实上，证明计算任务的 NP-完全性是阐明计算问题困难性的一个核心工具，这个工具被广泛应用于计算机科学及其他学科中。注意：NP-完全性不仅仅表明猜测一个问题是不可解的，它还表明了该问题的“丰富性”，即该问题足以用来对 NP 类中的其他问题进行“编码”。由于 NP-完全性的确切含义，在这里使用“编码”这个词是恰当的，而这又在不同计算问题之间建立起联系（无需考虑其“绝对”复杂性）。

上述对 NP-完全性的讨论中隐含了表示方法的重要性，因为其中提到了可以相互编码的不同问题。事实上，表示方法是复杂性理论的核心。一般地，复杂性理论研究那些在问题陈述（或问题的实例）中并不包含其解的问题。换言之，问题（或其实例）中包含了所有必要的信息，为了求解问题，只需对这些信息进行处理即可^①。因此，复杂性理论考虑对信息的处理，以及将问题从一种表示（其中给出了必要的信息）向另一种表示（求解的目标）的转化。其实计算问题的解只是给定信息的不同表示而已，即以更明确的方式表示问题的解。例如，关于给定布尔公式是否可满足的问题，其解并未包含在公式的表示中（而此时的计算任务就是使这一问题的解更加明确）。因此，复杂性理论澄清了关于表示的一个核心问题，即在一种表示方法中，明确信息与不明确信息之间的区别。此外，复杂性理论甚至还规定了关于不明确定性的量化等级。

一般来说，复杂性理论为以往人们思考的各种计算现象提供了一种新的视角，包括上面提到的问题的解、证明、表示方法以及诸如随机性、知识、交互、秘密以及学习等概念。下面将介绍这些概念，并讨论如何从复杂性理论的角度来理解这些概念。

随机性的概念长期以来一直困扰着思想家们。他们由存在性的角度来描述随机性：首先提出问题“什么是随机性”，再搞清楚它究竟是否存在（或世界是否是确定的）。复杂性理论则采取行为主义的方法：如果对象之间不能用高效的程序来区分，则将其定义为等价的。也就是说，如果不能预测掷硬币的结果，则将其定义为“随机”的（即使人们相信宇宙是确定的）。类似地，可以认为一个字符串（或字符串的分布）是“随机的”，如果不能区分其与均匀分布（不考虑是否能生成其中的字符）。有趣的是，用这种方法定义的随机性（或者更恰当地，伪随机性）可以被高效扩张，即在一个合理的复杂性假设（以下将讨论）下，可以把短的伪随机序列确定性地扩张为较长的伪随机序列。事实上，人们发现，随机性与不可解性密切相关。首先，注意到伪随机性的定义中涉及到不可解性（即区分伪随机的物体与均匀分布的物体是不可解的）；其次，如前所述，一个有关易求解但难求逆的函数（称为单向函数）的存在性的复杂性假设中，蕴含了存在确定性的程序（称为伪随机数发生器，Pseudorandom Generators），可以将短的随机数种子扩展为伪随机序列。由此可以认为，伪随机性发生器的存在性等价于单向函数的存在性。

复杂性理论中对于知识（有别于信息）的概念有其独特理解。具体来说，复杂性理论认为知识是艰苦计算的结果。因此，任何人能高效获得的信息都不被当作知识。特别是，利用可公开获取的信息，通过简单计算得到的结果不能称为知识。相反，利用可公开获取信息，

① 相反，在其他学科中，求解一个问题也许需要收集一些在问题描述中没有包含的信息。这些信息可以从辅助的记录中获取，也可通过新的实验得到。

但是经过大量复杂计算得到的结果可以称为知识，如果某人提供了这样的信息，那他就提供了“知识”。这又涉及到零知识交互的概念，零知识交互是指在交互过程中没有获得任何知识，但这种交互仍是有用的，因为它可以向一方证实某个预先提供数据的正确性。例如，可以通过交互式零知识证明令一方确信给定图的 3-可着色性，但是又不泄露具体的 3-着色方案。

上述讨论中提到了交互，其中将交互看作是获取知识和/或信任的工具。我们强调后一种应用，因为注意到当允许与证明者交互而不是仅仅阅读证明时，断言的正确性将更容易验证。换句话说，与一个好的老师交互比读书获益更多。我们认为，交互式证明的优势来自于随机化（即验证程序是随机化的），因为验证者的问题可以预先确定，而证明者只需提供交互的副本作为传统意义上的证明。

另一个与知识相关的概念是秘密：知识是指一方拥有而另一方没有（并且不能靠自身得到）的东西——因此，在某种意义上，知识就是秘密。复杂性理论与密码学关系密切，对于后者，一种粗略的理解——对易使用但难以误用的系统的研究。典型的密码系统涉及到秘密、随机性以及交互，同时也涉及到在易于合理使用与使系统偏离其预先定义行为的困难性之间的复杂性缝隙（Gap）。因此，密码学在很大程度上基于复杂性理论中的假设，其中代表性的研究成果将比较简单的计算原语（如单向函数）转化成为更复杂的密码应用（如安全加密方案）。

我们在前面讨论向老师学习与从书本中学习时曾经提及学习的概念。复杂性理论提供了前者更优越的证据，这在由公开信息中获取知识的语境中是成立的。相反，计算学习理论研究的是只能被学习者部分获取的学习对象（如基于几个随机点的函数值来重构一个函数，或者甚至是由学习者自己选择的点的函数值来重构函数）。复杂性理论清楚地指出了（计算学习理论中）学习的内在局限性。

复杂性理论研究各种计算任务。我们在前面提到过两种基本的任务类型：求问题的解（或者更确切地，“寻找答案”）和做出判断（如确定断言的有效性）。我们还暗示，在某些情况下这两种类型的任务是相关联的。现在考虑另外两种任务类型：对答案进行计数和生成随机答案。显然，后两种任务都至少与求相应问题的任意解难度相当，但是对于自然界中的某些问题，它们并没有明显地表现出更加困难。特别是在问题的某些正常条件下，近似地对解的数量计数以及生成近似随机的解，并不比找到任意一个解在难度上有明显的提高。

说到近似，我们注意到研究求“近似解”的复杂性也是很重要的。一种类型的近似问题涉及到定义在可能解集上的目标函数：近似的目的是为了求包含“几乎最优”值的解，而不是求达到最优值的解，其中对于“几乎最优”可以有不同的理解，从而得出不同水平的近似。有趣的是，在许多情况下，即使是对近似水平作了很大程度的放宽，求解时的难度也仍相当于解原始的（精确）搜索问题（即求近似解与求最优解难度相当）。不可思议的是，这些关于近似问题难度的结论与对概率可核查证明的研究相关，这种证明可以快速地进行概率验证。每一个证明都可以通过检查（证明中）常数个比特而高效地转化为容许概率验证的证明，这是令人惊奇的。现在回到近似问题，注意到在其他一些情况下，得到合理水平的近似解要比解原始的（精确）搜索问题更容易。

近似是对计算问题一种自然的弱化。另一种自然弱化是平均情况复杂性。这里的“平均”是指关于某种“简单”概率分布（代表了实际可能出现的实例的概率）取值。我们强调，虽然没有明确指出，迄今为止所有的讨论都针对着算法的“最坏情况”。最坏情况下的复杂度比平均情况下的复杂度更加健壮。初学者应该尽量避免这样一个有争议的问题，即何种实例是

“实际中重要”的，以及应该选择何种分布来分析平均情况。不过，对于平均情况复杂性也提出了比较健壮的理论，虽然其研究进展比不上最坏情况复杂性。

鉴于随机性在复杂性理论中的核心地位（这与在伪随机性、概率证明系统以及密码学研究中的作用一样显著），人们可能想知道，实际应用中需要的随机数是否能在现实中真正获得。一个引起普遍关注的问题是所谓“纯粹”的随机性（或“从坏的资源中提取好的随机性”）是否有可能。换句话说，是否可以利用“有缺陷”的随机源来构造几乎完美的随机源？对这一问题的答案当然取决于这种有缺陷随机源所使用的模型。对这一问题的研究也与复杂性理论相关，两者之间最紧密的联系存在于某些类型的随机数提取器与某些类型的伪随机数生成器之中。

至此，我们一直主要关注计算任务的时间复杂性，因为人们自然地会用时间来衡量效率。然而，时间并非唯一要考虑的资源。另一个重要资源是空间：在计算中需要（暂时）占用的存储空间。对空间复杂性的研究揭示了一些有趣的现象，这些现象似乎表明空间复杂性与时间复杂性之间存在本质的区别。例如，在空间复杂性中，对于断言（可以是任意一种特定类型）正确性证明的验证过程与验证同类断言的无效性证明具有相同的复杂度。

如果读者感到困惑，这也毫不奇怪。我们高速地飞越了一些山峰，感觉头晕目眩是很正常的。当然，本书的其余部分会提供完全不同的旅行体验。我们将从山脚开始，一步一步地爬山，并常常驻足来观察和思考。

绝对结论（亦称下限 Lower-Bounds）

坦率地说，复杂性理论中许多“大问题”（如最著名的 P-vs-NP 问题）的绝对结论至今尚且未知。然而，人们已经证明了几个显然是非平凡的结论。例如，人们证明了利用否定可以加速单调函数的计算（本来在计算过程中否定不是必需的）。另外，为了对计算过程进行低层次分析，人们引入并使用了许多有前景的技术。但是，在前言中也说明了，这些并非本书的重点。

1.1.2 复杂性理论的特征

成功来自于正确的抽象。

Avi Wigderson (1996)

一般而言，使用“正确的抽象等级”似乎是计算理论最主要的特征。正确的抽象等级意味着忽略那些次要的、依赖于上下文的细节，而使用能体现主要问题的定义（而不是把这些也忽略掉）。事实上，使用正确的抽象等级需要训练有素的良好判断力，而选择正确的抽象也属于研究成果的一种类型。

选择何种计算模型以及相应的复杂性测度和类型是计算理论研究中一个重要方面。通常认为应该理所当然地使用简单计算模型，并避免两种极端情况，即过于具体（从而涉及到太多细节），或过于抽象（造成含义不清）。一方面，（复杂性理论中使用的）主要的计算模型并不试图模仿（或反映）实际使用的计算机的特定操作。这将使复杂性理论的研究进展缓慢，并很难揭示本书中讨论的基本关系：大量的细节给理解造成障碍。另一方面，回避使用任何一种具体的计算模型（如回归函数论中的情况）也无益于复杂性理论中各种方法的引入及研究。事实上，正如我们将在 1.2.3 节看到的，人们一直选择简单的计算模型（不是实际使用计算机的映像），而避免由于模型的特殊性而造成的影响（通过密切观察各种模型）。研究在不

同计算模型（只要这些模型是“合理”的）下保持不变的复杂性类可以不受计算模型自身特征的影响。

复杂性理论的另一个重要方面是渐近分析（Asymptotic Analysis）的方法。具体而言，是将算法的复杂性当作其输入长度的函数，并研究该函数的渐近行为。人们发现隐藏于具体数量中的结构最终总会浮现。此外，还要视具体情况，根据不同的标准来定义函数。例如，在时间复杂性中，考虑的是对乘法运算封闭的函数类，而在空间复杂性中，则考虑在加法运算下封闭的函数类。无论何种情况，渐近分析都受所采用的复杂性度量影响。其实也可以创建一个不采用这些传统做法的复杂性理论，但是这样的理论只会更复杂。例如，我们不说求一个给定公式的可满足性赋值可在多项式时间内归约为对另一公式的可满足性判定，而是直接阐明该搜索问题与判定问题的复杂性之间的确切关系。

上述两个方面在计算理论的其他研究分支中也普遍存在。复杂性理论之所以特殊是由于其中涉及到计算理论中最基本的问题，即用何种方法来研究什么是可以高效计算的。复杂性理论的方法在自然界中较常见。其特点是主要关注相对效率（用相应的资源界限来体现）而不是具体的计算问题。许多情况下，复杂性理论研究并不针对某个特定计算问题或那些仅作为例证的问题。进一步来说，即使是研究特定计算问题，复杂性理论的研究目标（明确地或者至少暗示地）也只是理解某种资源界限内的计算局限性。

以上提到的一般方法似乎与该领域中概念的重要地位有关：要缜密地研究直观效率，必须从定义出发。由于研究针对任意可能的（实质性的）计算，所以不可能通过对某些具体现实的抽象（如一种具体的算法程序）来给出定义。定义应反映所有可能的现实，这意味着在给出定义时要依据概念性的原则，而不是仅凭经验。

1.1.3 本书内容概要

本书全面地介绍计算复杂性理论。全书包含 10 章和 7 个附录，既可作为教科书，也可供自学使用。前面的 10 章是本书的核心内容，写作风格类似于教科书。附录中提供了相关背景知识和一些附加内容，写法更像是综述。

1.1.3.1 全书内容组织

1.2 节和第 2 章提供了阅读本书其余章节所需的预备知识。从技术上说，本书其余章节中大量使用了这里出现的概念和结论。更重要地，1.2 节介绍了构成复杂性理论的概念性框架，为研究该领域中的问题及解提供了一个良好的视角。事实上，1.2 节和第 2 章是全书的基础，对复杂性理论有兴趣并且想进一步学习的读者必须理解这些内容。

本书其余部分则包含了更前沿的内容，如果只想对复杂性理论有一个最基本的了解，则这些知识不是必需的。特别是，虽然这些前沿性的章节之间有相互引用的情况，但其间的关系并不重要。因此，读者如果参考各个章节之间的关系图（图 1.1），则可以任意选择章节来阅读/讲授。当然，我们建议以书中的自然顺序阅读/讲授所有章节。

图 1.1 表明，在某些章（如第 3 章、第 6 章、第 10 章）中，将一些常常是独立介绍的专题放在一起，这样做与本书的研究方法有关。

至于附录，要注意某些附录（如附录 G 和附录 D、E 的部分内容）中提供了阅读前 10 章所需的基础知识。还有一些附录（如附录 B、C 及附录 D、E 的其他部分）则对章节的内容进行扩展（1.1.3.2 节将详细说明附录 A 和附录 F 中的函数）。

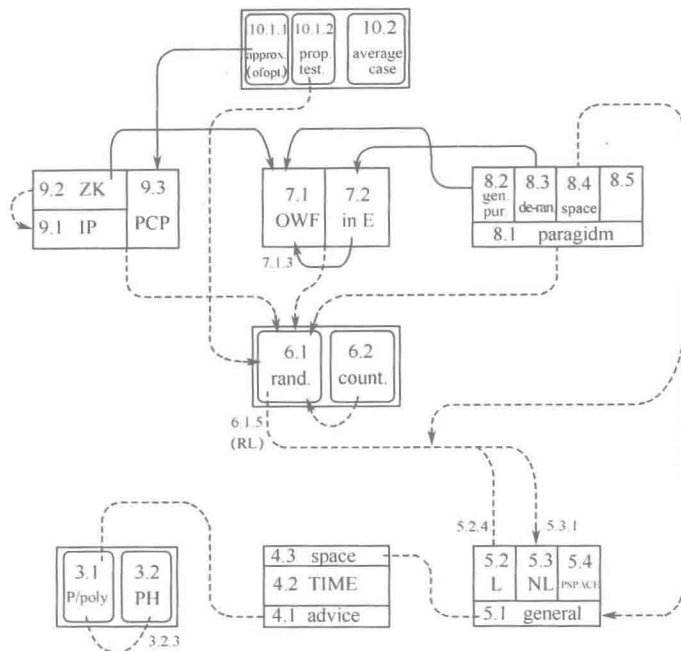


图 1.1 各章节间的关系。实心箭头表示使用了箭头指向部分的具体结论。虚线（及箭头）表示概念间存在重要联系；线条越粗，联系越紧密。当关系仅存在于小节之间时，标出了小节的编号

1.1.3.2 各章内容概述

以下给出各章及附录的内容概述。这一部分主要面向教师和/或专业人员，建议学生阅读前言中更浅易的概述。

1.2 节：预备知识。

本节介绍可计算理论的相关背景知识，是学习本书其余部分（以及全面学习复杂性理论的）的基础。最重要的是，本节讨论复杂性理论的核心概念，如搜索与判定问题，解这些问题的算法，以及算法的复杂度。另外，本节还介绍了计算的非一致性模型（如布尔电路、Boolean Circuits）

第2章：P、NP 和 NP-完全性。

本章介绍 P-vs-NP 问题的搜索形式及判定形式。另一个主要内容是 NP-完全性理论。此外，还介绍了（多项式时间）归约的一般概念，特别强调了自归约性。另外，本章讨论了 NP 中既非 NP-完全也不属于 P 的问题的存在性、最优搜索问题、coNP 类以及承诺问题。

第3章：P 与 NP 的变形。

本章介绍非一致的多项式时间 (P/poly) 以及多项式时间层级 (Polynomial-time Hierarchy, PH)。在定义这两者时采用了两种等价方法（如在定义 P/poly 时，既使用电路，也使用“接受建议的机器”）。另外，还证明了 NP 包含于 P/poly 中，从而 PH 可以收缩到其第二层（即 Σ_2 ）。

第4章：资源越多功能就越强大吗？

本章的核心问题是层级定理，该定理称更多的资源消耗可以解决更多的问题。证明时利用了界限函数，计算这些函数所使用的资源不能超过规定界限，否则就要使用缝隙定理 (Gap

Theorems)。

第 5 章：空间复杂性。

本章的主要结论包括检测（无向）图的连通性的对数空间算法、对 $\mathcal{NL}=\text{co}\mathcal{NL}$ 的证明，以及 $\mathcal{NL}$ -完全和 $\mathcal{PSPACE}$ -完全问题（分别在在对数空间以及对数时间归约下）。

第 6 章：随机性和计数。

本章主要讨论各种随机的复杂性类（如 $\mathcal{BPP}$ 、 $\mathcal{RP}$ 和 $\mathcal{ZPP}$ ）以及计数类 $\#P$ 。主要结论包括 $\mathcal{BPP} \subset P/\text{poly}$ 和 $\mathcal{BPP} \subseteq \Sigma_2$ 、求矩阵积和式（Permanent）的 $\#P$ -完全性、近似计数与均匀生成解之间的关系、近似计数到 $\mathcal{NP}$ 的随机归约，以及 $\mathcal{NP}$ 到具有唯一解的问题的随机归约。

第 7 章：困难性的用途。

本章讨论两个与 $P \neq \mathcal{NP}$ 相关的猜想。第一个猜想是 ε 中存在一些问题，不能用（非一致性的）小规模（即多项式规模）电路求解。第二个猜想与单向函数的概念等价。本章用大部分篇幅来解释“困难放大”结论，它可以将上述猜想转化为对 $\mathcal{BPP}$ 类（或许多密码学应用）进行非平凡去随机化的工具。

第 8 章：伪随机数发生器。

本章的重点是计算不可区分及伪随机性的概念。一般用途的伪随机数发生器（运行于多项式时间，并能经受多项式时间区分器的分析）被看作是通用模式的一种特例。本章还介绍了该模式的其他一些实例，包括对复杂性类如 $\mathcal{BPP}$ 类去随机化的发生器、可欺骗空间受限区分器分析的发生器，以及一些具有特殊用途的发生器等。

第 9 章：概率证明系统。

本章给出三种类型的概率证明系统：交互式证明（Interactive Proofs）、零知识证明（Zero-knowledge Proofs）和概率可检验证明（Probabilistic Checkable Proofs）。主要结论包括 $\mathcal{IP} = \mathcal{PSPACE}$ 、对任意 $\mathcal{NP}$ -集的零知识证明以及 PCP 定理。对于后者只简要介绍了两种已知的不同证明方法。

第 10 章：对复杂性要求的弱化。

本章提出两种类型的近似问题以及平均情况（或更恰当地，典型情况）复杂性定理。传统类型的近似问题与搜索问题有关，主要包含要求弱化后得到的标准优化问题。第二种类型是所谓的“性能测试”，主要包含要求弱化后的标准判定问题。平均情况复杂性定理涉及到几个非平凡的定义（如对分布类型的适当选择）。

附录 A：复杂性类汇总。

为本书中的大部分复杂性类给出自包含的定义。

附录 B：寻求下限。

第一部分是电路复杂性（Circuit Complexity），研究解决一般计算问题的（受限的）电路规模下限。第二部分是证明的复杂性（Proof Complexity），研究对同一自然定理不同证明的长度下限。

附录 C：现代密码学基础。

该附录的第一部分进一步阐述了单向函数、伪随机数发生器以及零知识证明（第 7 章～第 9 章）等基本工具。第二部分则利用这些工具研究了基本的密码学应用，如加密、签名，以及通用的密码协议。

附录 D: 概率论基础及随机性中的前沿问题。

概率论基础包括传统意义上的随机变量以及三个有用的不等式（即马尔可夫不等式、切比雪夫不等式以及切诺夫界）。前沿问题包括 Hash 函数的构造以及剩余哈希引理（Leftover Hashing Lemma）的各种不同变形，并对采样器与提取器（即随机性提取问题）进行了回顾与总结。

附录 E: 明确的构造。

讨论纠错码和扩张图的各种计算问题。关于纠错码，本附录中介绍了哈达玛达（Hadamard）码、Reed-Solomon 码、Reed-Muller 码以及具有常数码率和常数相对距离的二元码的构造。还简要回顾了局部可检测码（LTC 码）以及局部可译码的概念，以及一个列表译码界。在扩张图部分，本附录介绍了标准定义及性质，还介绍了扩张图的 Margulis-Gabber-galil 构造以及 Zig-Zag 构造。

附录 F: 一些省略的证明。

本附录中介绍一些证明，可以用来代替原始的和/或标准阐述。这些证明包括可以通过随机的 Karp-归约将 PH 归约到 $\#P$ ，以及 $IP(f) \subseteq AM(O(f)) \subseteq AM(f)$ 。

附录 G: 一些计算问题。

简要介绍了图论算法、布尔公式以及有限域。

参考文献

如 1.1.4.4 节所述，我们努力使参考文献列表尽可能短（但是仍达到了几百项）。为此省略了许多相关的文献。大体而言，本书在选择参考文献时更倾向于教科书和综述性文章。然而我们也尽量加入某个领域的重要论文。

本书未涉及的内容

前言中提到，本书并没有包含复杂性理论中的所有内容。被忽略的重要内容包括电路复杂性（Circuit Complexity，见参考文献[46, 236]）和证明复杂性（Proof Complexity，见参考文献[27]），在附录 B 中对这些内容有所提及。本书还省略了通常复杂性理论教程中涉及到的一些内容，包括分支程序和判定树（见参考文献[237]）、并行计算（见参考文献[141]），以及通信复杂度（见参考文献[148]）。最近 Arora 和 Barak^[14]编著的一本教材中包含了上述所有内容。最后，我们认为有两个研究领域与复杂性相关，虽然大多数人不认同这一点，这两个领域是分布式计算（Distributed Computing）^[17]和计算学习理论（Computational Learning Theory）^[142]。

1.1.4 写作方法与风格

通常人们认为，一本科学著作最重要的是其中包含的技术性结论，而讲述方法和动机则不是特别重要，它们只是为了进行“纠错”和/或让读者读起来舒服而已。人们甚至认为，与艺术作品一样，对科学工作的理解应该留给读者自行完成。

作者强烈反对上述观点，因为艺术与科学间存在着本质区别，而这种差别恰与每个工作的意义相关。科学工作关心的是内容（而非形式），而在探求真理的过程中，科学依据的是逻辑（而非艺术）。作者认为，对科学工作而言，要将重点放在所要解决的问题上，为什么要解决这个问题、对问题的叙述方式、获取答案的方法，以及答案中所体现的思想。阐明科学工作中的这些方面是很重要的。

在复杂性理论中，上述观点表现得尤为突出。首先，在复杂性理论中，概念起着至关重

要的作用（而在其他领域中并不总是如此，见 1.1.2 节）。另外，复杂性理论的概念部分内容异常丰富。因此，必须讲述这些内容，否则，就会漏掉该理论中最重要的部分。

不幸的是，概念相关的内容在其他书籍及/或综述中很少（明确的）讲述^①，从而导致一个（令人吃惊的）结果，就是学生们对于学到的定义和结论的意义及用途只有模糊不清的认识。我们认为，通过详细解释定义和结论的意义，可以轻而易举地避免这种结果。与此紧密相关的另一个问题是要使用“正确的”定义（能更好地体现概念的基本特征）并强调（由此产生的）“正确”的结论。本书在写作上正是体现了这种观点。

1.1.4.1 一般原则

与上述观点一致，本书的要点是概念。每当要描述一个对象时，由与之相关的直观问题出发，解释这些问题的重要性，其所特有的叙述方式（即实际选择的形式化方式），获取答案的方法，以及这些答案中所体现的思想。因此，正文中用了较多篇幅来讨论实际的定义和结论背后所体现的概念及思想。

本书的材料组织围绕着概念性主题，体现了基本概念和/或普遍性问题。我们较少涉及到特定的计算问题，除非是利用它们来澄清概念或者该问题恰好能用于解释一般的概念。例如，在本书中，具体的“完全问题”（如 NP-完全问题）与它们在哪一方面达到“完全”相比总是居于次要地位。^②

1.1.4.2 几个特殊做法

不同于其他教科书，本书的写法比较特殊，这是出于概念上的考虑。有时这种差异可以导致技术上的简化。本小节只介绍一般性的题目和/或对大部分表述具有全局性影响的特征。主要面向教师和/或专业技术人员。

避免使用非确定型的机器

我们将尽可能地避免使用非确定型机器。在本书几个不同地方（如 2.1.5 节），我们认为这些假想的“机器”从概念和技术的角度来看都起着负面作用。非确定型机器对概念的影响在于为什么要关心这类机器能做什么，这一点尚不十分清楚。当然，考虑这些问题的原因是清楚的，特别是当这些假想的“机器”无法提供一种（方便但投机取巧的）叙述基本问题的方法时。非确定型机器技术上的影响在于它们会使学生困惑。另外，它们并没有提供处理更前沿问题（如计数类）的最佳方式。

与此相反，我们在大部分表述中以搜索问题为基础。具体来讲，由具有可高效验证的解的搜索问题构成的 PC 类（见定义 2.3），在正文中占据重要地位。事实上，对于这类问题的定义比 NP 的标准定义（基于非确定型机器）要稍微复杂一些，但是随着讨论的进一步展开，以 PC 类为核心在技术上的优点将逐渐增加。当然， PC 类是一种基本的计算问题类，而这也正是采用 PC 类的主要动机。（事实上，对于 P-vs-NP 问题，有一种在概念上更吸引人的陈述方法：问是否 PC 中每个搜索问题都可以高效地求解。）

① 猜测这种现象的成因很有意思。一种猜测是为了解释复杂性理论的概念内容需要从技术上直接做出大胆的哲学断言，而与哲学的这种联系和大多数复杂性理论的研究者的习惯不太一致。

② 我们承认，一个自然的计算问题可以诱导出一类与其在计算上等价的问题，此时，这一类问题与原始问题相比也许并不那么有趣。对于本书中的任何复杂性类，情况并非如此。然而，在某些情况下（如 NP 和 $\#P$ ），历史演化实际上是从一个特定的计算问题出发，扩展到计算上等价的一类问题。然而，在本书中出现的所有情况下，通过对文献的回顾与总结，可以发现最终得到的类实际上远比原始问题更有意义。

避免计算模型的影响

复杂性理论涉及到高效计算的概念。这一概念的严格定义参照了某些具体的计算模型；然而，本书涉及的所有问题及答案都无关于具体模型的选择，当然，模型必须是“合理的”（毫无疑问，这属于直觉问题）。上文反映了在做出严格定义与期望独立于技术选择之间的冲突，这是不可避免的。另外还体现了一个事实，即从本质上讲，我们讨论的问题都与模型无关（即独立于各种技术上的选择）。注意到，我们并没有否认依赖于模型的问题的存在性，但会极力避免讨论这些问题，并认为其无关紧要。与通常想法相反，上述说明不仅针对时间复杂性，也针对空间复杂性。然而，无论是哪种复杂性，当资源濒于临界时（如线性时间或亚指数的空间），与模型的无关性也许不成立。

但是在某些情况下，我们也可能会选择特定问题。特别是与多项式时间复杂度相关的效率问题（见 1.2.3.4 节）。事实上，与高效计算相关的所有问题及其解也可以不根据多项式时间复杂度来分类（即可以明确地写出问题复杂度之间的函数关系式），然而这个想法很难实现。

1.1.4.3 技术细节

一般情况下，技术上越复杂，就意味着需要使用更多层次的阐述（从最高层开始，在必要时加入多个层次的细节）。特别地，本书对于不太简单的证明，我们会试着先阐明关键思想，而将具体细节推迟给出。我们还将清晰地指出从高层描述到具体实现细节之间的段落关系（如“细节如下”）。在某些情况下，特别是对于一些前沿性结论，只给出证明梗概，读者可以自行补充剩余的细节。

本书中绝大多数结论都提供了证明。某些情况下只给出了证明思想或证明概要，而不要读者阅读过于难懂的细节（这种情况书中一般都会明确指出）。其中一个典型的例子是 PCP 定理（定理 9.16）的证明。

在我们看来，复杂性理论中一些内容既非该领域中的“结论性”内容，也不代表“正确”研究方法，我们尽可能避免这样的内容。因此，在本书中可能看不到一些最“著名”的结论。

1.1.4.4 组织原则

每个主要章节都从一个高度概括的综述开始，以章节注释和习题结束。设计习题的主要目的不是测试或者激发创新，而是为了帮助读者进一步理解正文内容。部分习题涉及到进一步的学习内容（给出了足够多的提示）。

本书囊括了从目前的本科生课程（可计算性和复杂性理论基础）到为高年级研究生开设的专题。然而，这种状况在将来也许会有所改变（希望如此），从而使本科生更容易接触到更多的复杂性理论知识。我们相信，相比于只阅读基本章节（如 1.2 节及第 2 章）的读者而言，能阅读更前沿章节的读者群具备更深厚的专业背景，这是毫无疑问的。因此，与第 2 章相比，后续各章的表述方式将更专业化。

如前所述，本书重点讨论复杂性理论的高层次研究方法，而对于较低层次的方法（如下限等）只在附录 B 中作简单介绍。其他附录内容与复杂性理论紧密相关但又并非其有机组成部分（如密码学基础）。^①1.1.3 节进一步介绍了各个章节及附录。

为了控制参考文献列表的长度，我们省略了许多相关的文献。主要采用的技巧是间接引用其他资料中的文献，特别是当这些资料很重要而不得不引用时。事实上，我们在选择参考

^① 7.1 节将进一步阐明，我们建议在复杂性理论课程中不包含密码学的基础内容。事实上，密码学被称为是复杂性理论最具吸引力的应用，但在本课程中对密码学的浅显介绍（从复杂性理论角度）可能会误导读者。

文献时更偏重于教科书及综述性文章，这些文献为进一步深入学习提供了最佳途径。当然我们不会省略某个领域的关键论文。某些情况下，当需要某个相关结论的出处而又不能采用上述技巧时，也会引用那些不太重要的论文。

作为一种策略，我们尽量避免在正文中出现参考文献引用。极个别的例外是被省略细节的出处，以及非常著名的术语（以研究者名字命名）的出处。一般情况下，每一章的注释中都包含了相关参考文献的引用。

教学注释：正文中还包含一些教学注释。这些注释提供了相当有价值的建议和/或对正文的解释。

1.1.4.5 恳请容忍

本书在编写时尽量尝试着满足各种读者的需求，从对复杂性理论一无所知的外行到该领域的专业人士。这体现在，在本书的每一部分，都对表达方法进行了相应的调整，使其适合于具备很少背景知识但又愿意阅读该部分章节的读者。然而，在极少数情况下，不可避免地会包含仅仅针对专业人士的前沿性内容。因此，具有更多背景知识的读者可以跳过某些细节，而背景知识欠缺的读者则可以忽略掉某些前沿性内容。为便于阅读，文中加入了足够多的标记，但在许多地方，还希望读者能自己做出判断（并容忍一个事实，即可能要求读者投入精力来适应其他类型读者的阅读兴趣）。

我们强调，本书的不同部分确实针对着不同类型的读者。具体而言，1.2 节和第 2 章主要针对没有复杂性理论（甚至是可计算理论）背景的读者，而后续章节则假设读者具备这些基础知识。除了要熟悉基础知识，更前沿的部分还假设读者掌握了更多的专业知识。

1.1.4.6 关于写作动机的附加说明

作者猜测有人会批评我们对一些从技术上讲不太重要的问题讨论的篇幅过长。事实上，大多数研究者会自动忽略掉各种概念讨论，认为这些内容不重要，重要的是，在技术上富于挑战性的内容。结果学生掌握了技术细节，却弄不清楚其含义。因此，作者建议学生们多花点时间在概念上，虽然有些学生认为概念可以忽略。

本书中关于动机的讨论不一定代表了研究者研究该领域和/或做出重要贡献的最初动机。相反，这些动机只是作者从概念角度出发而提出的。

1.1.5 标准符号及习惯性用法

以下是本书的常用符号及习惯用法。

标准渐近符号

对于整数函数，我们采用标准渐近符号，即对于函数 $f, g: \mathbf{N} \rightarrow \mathbf{N}$ ，如果存在常数 $c > 0$ ，使得对任意 $n \in \mathbf{N}$ ，有 $f(n) \leq c \cdot g(n)$ （或 $f(n) \geq c \cdot g(n)$ ），则记作 $f = O(g)$ （或 $f = \Omega(g)$ ）。通常用符号“poly”表示一个未定义的多项式，用 $f(n) = \text{poly}(n)$ 表示“存在一个多项式 p 使得对任意 $n \in \mathbf{N}$ ，有 $f(n) \leq p(n)$ ”。我们还用 $f = \tilde{O}(g)$ 表示 $f(n) = \text{poly}(\log n) \cdot g(n)$ ，用 $f = o(g)$ （或 $f = \omega(g)$ ）表示对任意常数 $c > 0$ 以及任意足够大的 n ，有 $f(n) < c \cdot g(n)$ （或 $f(n) > c \cdot g(n)$ ）。

取整问题

通常我们会忽略取整问题，这意味着可能会假设 $\log_2 n$ 是一个整数，而不使用更复杂的

形式: $\lfloor \log_2 n \rfloor$ 。类似地, 我们会假设各种等式存在整数解 (如 $2^n = m^m$), 即使这一点根本不可能成立 (如 $2^n = 3^m$)。在所有情况下, 只需考虑近似满足这些等式的整数即可 (并处理由于近似而产生的问题, 本书中这类问题常被忽略)。

标准的组合论及图论术语及符号

对任意集合 S , 用 2^S 表示 S 的幂集 (即 $2^S = \{S' : S' \subseteq S\}$)。对于自然数 $n \in \mathbb{N}$, 定义 $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$ 。本书提到的许多计算问题涉及到有限 (无向) 图。这样的图记作 $G = (V, E)$, 其中 V 表示顶点 (Vertices) 集合, E 表示边 (Edges) 集合, 由一些无序的顶点对组成。一般默认为图是无向的, 而有向图 (Directed Graphs) 中包含了顶点和有向边集合, 其中有向边用有序的顶点对表示。我们还会用到图论中的其他术语如连通性、无圈性 (即不含简单圈)、树 (连通且非循环)、 k -可着色性等。附录 G.1 中进一步介绍了图论知识及与图论相关的计算问题。

排版习惯

用斜体字表示严格定义的复杂性类 (如 $\mathcal{NP}$), 但是只有当给出了定义之后才这样写。另外, 为了保持一类问题关于特定表述的模糊性, 使用罗马字体 (如 NP 既表示搜索问题也表示判定问题)。类似地, 用打字型字体 (Typewriter) 表示已被正式定义的计算问题, 而用斜体字 (Italic) 表示一般问题及算法。

1.2 计算任务及模型

你可以说, 我们让你谈谈女人和故事——
但是这与自己的房间有什么关系呢? 我试着解释一下。

弗吉尼亚·伍尔夫, 自己的房间

本章给出阅读本书其余部分所需的预备知识, 换言之, 讨论计算任务的概念, 并构造解决这些任务的计算模型。我们先引入计算任务 (或问题) 的通用框架: 该框架给出了对问题实例的描述 (为二元序列), 并重点讨论两种类型的计算任务 (即求解和做出判定)。为了便于研究解决这些任务的方法, 后者的定义中涉及到无限多种可能的实例 (每一个均为有限的对象)。^①

给出了计算任务的定义之后, 我们就转向研究解决这些任务的方法, 这用某种计算模型术语描述。计算模型是本节的主要内容。具体而言, 我们考虑两种类型的计算模型: 一致性模型和非一致性模型。一致性模型对应于算法的直观概念, 本书的其余章节都将建立在这个模型之上 (重点是高效的算法)。与此相反, 非一致性模型 (如布尔电路, Boolean Circuits) 更便于深入观察计算的进展, 因此本书偶尔也会使用这种模型。

本小节内容组织

1.2.1 节~1.2.3 节讲述传统的可计算性课程内容, 然而与传统课程相比, 我们的侧重点有所不同。特别是, 强调了通用机器的概念 (1.2.3.3 节), 解释了高效计算与多项式时间算法之

^① 对于不同方法的比较似乎需要考虑无限多种可能的实例, 否则, 对描述语言的选择将占主导地位并可能将讨论引入歧途 (见 1.2.3.4 节中对 Kolmogorov 复杂性的讨论)。

间的关系(1.2.3.4节),并定义了预言机(Oracle Machines)(1.2.3.5节)。这些内容(除柯尔莫果洛夫复杂度 Kolmogorov Complexity 之外)在本书其余章节中常常被引用。而1.2.4节介绍的非一致性计算模型(即各种类型的布尔电路),在其余章节中则很少提到(我们还希望读者关注一下1.2.5节中对一般复杂性类的讨论)。因此,1.2.1节~1.2.3节(以及1.2.5节)属于阅读本书必备的基础知识,而1.2.4节则不在此列。

教学注释: 作者认为可计算性(即讨论什么可以被计算而非被有效计算的课程)并不需要整整一个学期的时间来学习。相反,本科生应该学习计算复杂性理论的课程,而其中包含可计算性的内容作为基础。具体而言,可计算性内容最多占课程学时数的25%,而计算复杂性课程应将重点放在基本的复杂性问题上(如P、NP以及NP-完全性),并有选择地介绍一些前沿知识。事实上,本课程的基础是本书的第1章和第2章(以及其他章节中的部分内容)。

1.2.1 表达方式

在数学及其他相关学科中,一种习惯做法是直接讨论某个对象而不规定具体的表达方式。然而,在计算理论中,这是行不通的,因为此时对对象的描述方法至关重要。从某种意义上说,计算,仅仅是将对同一对象的一种表达转化为另一种表达而已。特别是被设计用于解某个问题的计算,仅仅是将问题实例转化为问题的解,而后者则(可能部分地)被看作是对问题实例的一种表达。事实上,任何一个充分定义的问题中都暗示了该问题的解,而计算的任务是让这个解更明确。

计算任务针对的是那些以某种规范方式表达的对象,即用一种“明确的”且“完整的”(但并不“过分”)方式来描述相应的对象。我们将只考虑有限对象,如数、集合、图以及函数(并坚持区分这些不同类型的对象,虽然实际上它们都是等价的)。对于数、集合和函数的描述都相当直截了当,而对图的描述可以参考附录G.1。

串: 以有限二元序列表示的有限对象,称为串(String)。对于自然数 n ,用 $\{0,1\}^n$ 表示所有长为 n 的串,以后都称为 n 比特(长)串。所有串的集合记作 $\{0,1\}^*$,即 $\{0,1\}^* = \bigcup_{n \in \mathbb{N}} \{0,1\}^n$ 。对于 $x \in \{0,1\}^*$,用 $|x|$ 表示 x 的长度(即 $x \in \{0,1\}^{|x|}$),通常用 x_i 表示 x 的第 i 个比特(即 $x = x_1 x_2 \cdots x_{|x|}$)。对 $x, y \in \{0,1\}^*$,用 xy 表示串 x 与 y 的级联。

有时我们会将 $\{0,1\}^* \times \{0,1\}^*$ 合写为 $\{0,1\}^*$,读者可以只考虑一种充分的编码(即将一对序列 $(x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n) \in \{0,1\}^* \times \{0,1\}^*$ 编码为串 $x_1 x_2 \cdots x_m 01 y_1 y_2 \cdots y_n \in \{0,1\}^*$)。类似地,我们可能会将由(固定长度或可变长度的)串构成的序列直接表示为一个串。因此,当需要强调这样一个序列(或其他对象)被当成一个单独的对象时,使用符号 $\langle \cdot \rangle$ (即“把 (x, y) 编码为串 $\langle x, y \rangle$ ”)。

数: 除非明确指出,否则,自然数都被编码为二进制形式,即用串 $b_{n-1} \cdots b_1 b_0 \in \{0,1\}^*$ 表示整数 $\sum_{i=0}^{n-1} b_i 2^i$,典型地,我们假设在这种表示中首位不是0(即 $b_{n-1}=1$)。有理数可以表示为一对自然数。在少数情况下,计算问题的部分输入可能是实数,此时,将这些实数用有理数来近似代替。

特殊符号: 用 λ 表示空串(即 $\lambda \in \{0,1\}^*$ 且 $|\lambda|=0$),用 ϕ 表示空集。有时使用一些不属于

$\{0,1\}^*$ 的特殊符号会很方便, 如符号 $\perp$ 通常用于指示 (由某些算法产生的) 错误。

1.2.2 计算任务

搜索问题和判定问题是两种基本的计算任务。在这两种类型中, 一个关键概念是问题实例以及对问题的说明。

1.2.2.1 搜索问题

搜索问题为每个实例规定了有效解的集合 (可能是空集)。即给定一个实例, 要求找到相应的解 (或者判定不存在这样的解)。例如, 解方程组问题。毫无疑问, 计算机科学中许多问题都与求解各种搜索问题相关 (如找到图中的最短路径, 为一组数排序, 在给定串中寻找给定样品等)。此外, 搜索问题对应于人们日常所理解的“解一个问题” (如寻找两个地点之间的路径), 因此讨论解决搜索问题的可能性及复杂性符合大多数人的经验。

以下关于解搜索问题的定义中, 潜在的解决者是一个函数 (可以看作是一种解决策略), 可能的解集与每个实例相关, 而实例被“压缩”为一个二元关系。

定义 1.1 (解搜索问题) 设 $R \subseteq \{0,1\}^* \times \{0,1\}^*$, $R(x) \stackrel{\text{def}}{=} \{y: (x,y) \in R\}$ 表示实例 x 的解集, 称函数 $f: \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ 解搜索问题 R , 对任意 x : 如果 $R(x) \neq \emptyset$, 则 $f(x) \in R(x)$, 否则, $f(x) = \perp$ 。

事实上, $R = \{(x,y) \in \{0,1\}^* \times \{0,1\}^* : y \in R(x)\}$, 而只要问题有解 (即集合 $R(x)$ 非空), 则要求 f 找到一个解 (即给定 x , 输出 $y \in R(x)$)。另外, 还要求 f 不能输出错误的解 (即如果 $R(x) \neq \emptyset$, 则 $f(x) \in R(x)$, 否则, 如果 $R(x) = \emptyset$, 则 $f(x) = \perp$), 这反过来又意味着 f 指示着实例 x 是否有解。

一个有趣的特例是存在唯一解 (对所有可能的实例) 的搜索问题, 就是说, 对任意 x , $|R(x)| = 1$ 。此时, R 本质上成为一个 (绝对的) 函数, 而解 R 的搜索问题意味着计算函数 R (或对 R 求值) (或者可以用函数 R' 表示, 定义为 $R'(x) \stackrel{\text{def}}{=} y$ 当且仅当 $R(x) = y$)。人们熟知的例子包括为一组数排序、整数相乘、求合数的素因子分解等。

1.2.2.2 判定问题

判定问题规定了可能的实例子集。给定一个实例, 要求判断这个实例是否属于规定的子集中 (如素数集、连通图集或有序的数集等), 如整数的素性判定。与上述搜索问题相对应的一个重要特例是有解的实例集合。即对任意二元关系 $R \subseteq \{0,1\}^* \times \{0,1\}^*$, 考虑集合 $\{x: R(x) \neq \emptyset\}$ 。事实上, 能够判定是否有解是解相应搜索问题的先决条件 (由定义 1.1)。一般来说, 判定问题是指做出二元判断这种自然的任务, 这在我们的日常经验中并不陌生 (如判断交通指示灯是否为红灯)。在以下关于解判定问题的定义中, 无论何时, 潜在的解决者仍是一个函数, 不过此时是布尔 (Boolean) 函数, 要求其对给定集合做出成员判定。

定义 1.2 (解判定问题) 设 $S \subseteq \{0,1\}^*$, 称函数 $f: \{0,1\}^* \rightarrow \{0,1\}$ 解 S 的判定问题 (或对 S 作出成员判定), 如果对任意 x , $f(x) = 1$ 当且仅当 $x \in S$ 。

通常将 S 的判定问题当作 S 本身, 而将 S 用其特征函数 (即函数 $\chi_S: \{0,1\}^* \rightarrow \{0,1\}$, $\chi_S = 1$

当且仅当 $x \in S$) 表示。注意: 当 f 能解搜索问题 R 时, 定义布尔函数 $f': \{0,1\}^* \rightarrow \{0,1\}$, $f'(x) \stackrel{\text{def}}{=} 1$ 当且仅当 $f(x) \neq \perp$, 则 f' 可以解 $\{x: R(x) \neq \emptyset\}$ 的判定问题。

思考

大多数人认为搜索问题要比判定问题更自然: 典型的情况是, 人们常常愿意寻找问题的解, 而不是满足于知道问题是否有解。当然, 搜索问题的重要性一点也不亚于判定问题, 只是对其研究需要更复杂的方法。这是把判定问题作为重点的主要原因。本书则尝试将搜索问题也置于较重要的地位。

1.2.2.3 承诺问题 (一个前沿性注释)

许多搜索问题和判定问题都可以更自然地利用承诺问题术语来解释, 承诺问题的实例是 $\{0,1\}^*$ 的一个子集而非 $\{0,1\}^*$ 本身。特别地, 注意到许多搜索及判定问题的形式化定义都涉及到某种特定类型的实例 (如一个方程组、一对数字、一个图), 而这些对象只使用 $\{0,1\}^*$ 的某个严格定义子集描述。目前为止, 我们忽略了这个问题的, 但是在 2.4.1 节中会再次提到。这里只需注意, 在一些典型情况下, 可以假设每个串代表某个合法对象, 从而忽略该问题 (例如, 对于那些在对象的自然描述中没有出现的串, 可以假设其代表着某个固定的对象)。

1.2.3 一致性模型 (算法)

科学是一致的。

Laci Lovász (根据 Silvio Micali, 1990)

最后讨论本节 (1.2 节) 的核心问题, 即一致性计算模型。我们对计算机十分熟悉, 会用计算机程序来处理数据。这种熟悉根植于计算较积极的一面, 即从经验上人们认为计算机能做一些事情。相反, 复杂性理论重点关注什么是计算机不能做的, 或者对计算机能做与不能做的事情进行区分。为了实现这种区分, 需要准确定义所有可能的计算过程, 即我们应该对所有可能的计算过程建立一个清晰的模型 (而不仅仅是熟悉某些计算过程)。

1.2.3.1 概述与通用原则

在形式化定义之前, 我们先给出一个通用的抽象描述, 目的是帮助理解所有人为的和自然的过程。事实上, 人为的过程与计算机相关联, 而自然的过程在这里是指自然现象 (物理、生物甚至社会现象) 的 “机械化部分” (尝试对其建立模型)。

所谓计算, 是指通过重复使用一种预先制定的规则来改变环境的过程。这里一个关键性限制条件是规则必须是简单的: 在每次应用时, 只依赖于且只影响环境的一 (小) 部分, 称为激活区 (Active Zone)。我们可以对比一下先验有界的激活区 (以及修改规则) 与先验无界的整个环境。注意到虽然每次使用规则时只产生极其有限的影响, 但多次使用之后, 结果将异常复杂。换言之, 计算会以一种十分复杂的方式来修改与之相关的环境, 虽然它只是重复使用简单规则的过程。

前面暗示了, 可以用计算这一概念对自然现象的 “机械化” 部分建立模型, 就是说, 确定现实 (而非现象在某个特定时刻的特殊状态) 演变的规则。此时, 研究的出发点是自然现象的实际演化过程, 而研究的目的是找到这个自然过程背后的 (计算) 规则。在某种意义上, 科学研究的目标总体上可以描述为寻找支配各种自然现象 (或者这些现象的抽象) 的 (简单)

规则。

然而，我们讨论的重点是由人类设计的计算规则，借此可以对相应的人造环境产生特定的期望效果。因此，我们将从一个期望的功能出发，目标是设计影响这个功能的计算规则，这种计算规则被称为算法。大体而言，算法是指用高级语言编写的计算机程序，以下将详述。

我们感兴趣的是计算过程（或算法）将使环境产生何种变化。在本书的（大部分篇幅）中，我们假设，当计算在任意一个有限的初始环境中被激活时，它将在有限步之后中止。通常，初始环境对一个输入串编码，而终止环境（即计算结束时的环境）对输出串编码。我们考虑由计算所诱导的输入到输出间的映射关系，这是指对于每个可能的输入 x ，考虑计算停止时输出的 y ，此时称计算将输入 x 映射到输出 y 。因此，一个计算规则（或算法）确定了一个（由其计算的）函数：该函数恰好就是上文所说的由输入到输出的映射。

在本书其余部分（即除本章之外），我们还会考虑计算对于每个可能输入运行的次数（即应用规则的次数）。这也可以看作是一个函数，称为计算过程（或算法）的时间复杂度（Time Complexity）。虽然时间复杂度是针对每个输入而定义的，但人们通常考虑的是对于每种输入长度而言，取具有相同长度的输入的最大时间复杂度。

为了严格地定义计算（及计算所用的时间），需要指定计算模型，即对于计算环境及可应用于该环境中的一类规则提供具体定义。这种模型对应着实际中计算机（PC 机、大型机或计算机网络）的抽象。常使用的一个简单抽象模型是图灵机（Turing Machines）（1.2.3.2 节）。因此，算法通常由相应的图灵机（算法的时间复杂度则由相应图灵机的时间复杂度来表示）来形式化定义。然而，在这里要强调，计算理论中的许多结论，并不考虑特定的计算模型，只要模型是“合理的”（即满足前面提到的简单条件，并可用于完成某些简单计算任务）即可。

计算的对象是什么？

上述讨论中暗示了，提到的算法（即计算的过程）其实是计算函数的方法。具体而言，定义函数 $f_A: \{0,1\}^* \rightarrow \{0,1\}^*$ ， $f_A(x) = y$ ，如果对于输入 x ，算法 A 输出 y 并停止，则称 A 能计算函数 f_A 。然而，算法还可用于“解搜索问题”或“作出判断”（如定义 1.1 和定义 1.2）。我们称算法 A 解 R 的搜索问题（或对 S 进行成员判定），如果 f_A 能解 R 的搜索问题（或能判定 S 中成员）。以下将算法 A 与由 A 计算的函数 f_A 相关联，即用 $A(x)$ 来代替 $f_A(x)$ 。为了便于将来参考，将前面的讨论概括如下：

定义 1.3（用于求解问题的算法） 用 $A(x)$ 表示算法 A 对于输入 x 的输出，称 A 解搜索问题 R （或判定问题 S ），如果作为函数的 A 能解 R （或 S ）。

1.2.3 节的剩余内容组织

1.2.3.2 节中给出了图灵机模型的粗略描述。这只是为研究计算及其复杂性而提供一个具体模型，本书中大部分内容将不依赖于这个模型的细节。1.2.3.3 节及 1.2.3.4 节中介绍了合理的计算模型所具备的两个基本性质：不可计算的函数的存在性以及通用计算的存在性。1.2.3.5 节中定义了计算的时间（及空间）复杂度。本节还介绍了预言机及有限制的计算模型（分别见 1.2.3.6 节及 1.2.3.7 节）。

1.2.3.2 一种具体模型：图灵机

图灵机模型为算法的形式化提供了一个相对简单的工具。由于该模型的简单性而使解决实际问题的机器的设计变得格外复杂，但是分析起来却更简单。由于复杂性理论的核心是对

机器的分析而非设计，所以选择图灵机模型是恰当的。我们再次强调这个模型仅仅用于直觉上算法概念的形式化，实际上，人们更关心的是直觉概念而非其形式化定义。特别地，本书提到的所有结论对于算法概念的其他“合理的”形式化定义都成立。

图灵机模型的目标不是为实际使用的计算机提供一个准确的（或紧凑的）模型，而是为了理解其内在的局限性及能力（即一个计算任务可以由实际计算机完成当且仅当其他可以由图灵机完成）。与实际使用的计算机相比，图灵机模型是过于简单了，它在抽象时回避了许多计算机在实际应用时的重要问题。然而，这些问题与复杂性理论所要解决的高层次问题无关。事实上，为获得良好的实践，不仅要求人们具备比良好理论更精准的理解，而且首先还必须提供良好的理论。

历史上，图灵机模型出现于现代计算机诞生之前，其目的是为计算提供一个具体模型，并定义可计算的函数。^①事实上，这种具体模型澄清了可计算函数的基本性质，并在定义可计算函数的复杂性方面起到了关键作用。

图灵机模型被看作是对人在演草纸上进行的数学计算过程的一种抽象。在每一步计算中，人眼看到纸上的某个位置，根据所看到的和脑中所想到的来修改这个位置的内容，然后目光转移到一个相邻位置。

实际模型

以下给出图灵机模型的高层次描述，有兴趣的读者可以参考标准的教科书（如参考文献[208]）来获知更多细节。回忆一下，我们需要规定可能的环境集合、机器（或计算规则）集合以及这种规则作用于环境时产生的影响。

- 图灵机的运行环境主要由一个无限长的单元序列组成，每个单元能存储一个符号（即有限集 $\Sigma \supset \{0,1\}$ 中的一个元素）。机器在运行时，从单元序列的最左边开始，逐渐向右移动（图 1.2）。另外，运行环境中还包含了机器在序列上的当前位置，以及机器的内部状态（是有限集 Q 中的一个元素）。上述单元序列称为带（Tape），其存储内容、机器的当前位置及内部状态合称为图灵机的当前配置（Instantaneous Configuration）。

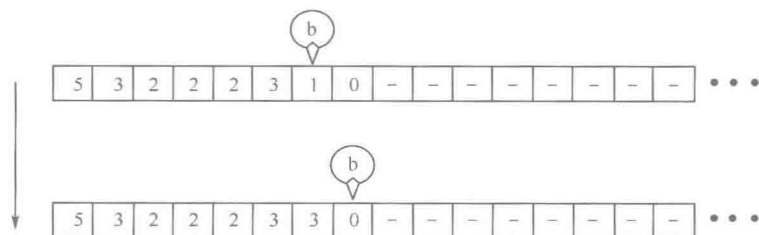


图 1.2 图灵机的一步运算

- 图灵机本身主要是一组有限的规则（即一个有界函数）构成，称为迁移函数，该函数的定义域是所有可能的符号—状态对。具体地，迁移函数是从 $\Sigma \times Q$ 到 $\Sigma \times Q \times \{-1, 0, +1\}$ 上的映射，其中 $\{-1, 0, +1\}$ 指示机器的行为（“左移”“右移”或“保持不变”）。另外，描述机器时还应该规定初始状态及停机状态，当机器到达停机状态时计算停止。^②

① 相反，由“递归函数”的抽象定义出发，无需参考任何的计算模型（而是基于一个直觉，即任何一种模型都应该支持递归函数的合成），便可得到一类“可计算的”函数的定义。

② 可以将带想像成如图 1.2 所示，我们还采用这样一个习惯：如果机器试图向带的最左端移动，则认为已经停机。

我们在这里强调，与机器的有限描述相反，带的长度是无限的（它可以被简单地看作是无限长的带）。

• 这样一个图灵机的一步计算依赖于其当前位置、该位置存储的内容以及机器的内部状态。根据后两个因素，迁移函数确定一个新的符号-状态对，并确定一个行动指示（“左移”“右移”或“保持不变”）。机器相应地修改原存储单元中的内容及内部状态，并按照指示的方向移动。假设机器当前状态为 q ，当前位置处存储内容为符号 σ ，并假设迁移函数将 (σ, q) 映射为 (σ', q', D) ，则机器将当前存储单元中的内容修改为 σ' ，状态修改为 q' ，然后向方向 D 移动一个单元。图 1.2 中表示了图灵机的一步运算，当前状态为“b”，当前位置处的存储内容为二元符号 σ ，将符号 σ 用 $\sigma+2$ 代替，维持原有状态，并向右移动一格。^①

我们形式化地定义后继配置函数（Successive Configuration Function）为图灵机运行一步时，将当前配置映射到结果配置的函数。如上所述，这个函数只是对图灵机的配置稍做改动，即最多只有一个存储单元的内容被修改（即机器的当前位置处），此外，机器的状态及位置也可能会改变。

图灵机的初始环境（或配置）是指机器停留在第一个位置（最左边）处并处于初始状态。在初始配置中，通常还强制性地令带上的存储单元有前导比特，这些比特的级联就是图灵机的输入。带上的其他位置处有一个特殊符号（在图 1.2 中记作“-”）。一旦停机，就将机器停机位置左边单元中（在停机时刻）的内容作为输出。^②因此，每个图灵机定义了一个将输入映射为输出的函数，称为由该图灵机计算的函数。

多带图灵机

在大部分叙述中，位置是指“机器头部”在带上的位置（而不是“机器在带上的位置”）。对这个基本模型进行推广时，采用标准术语将更直观，这里的“推广”是指将单带图灵机推广为支持常数数量个带的图灵机模型。在所谓的多带机（Multi-tape Machines）模型中，机器在每个带上都有一个头部位置，每一步运行取决于机器在每个带上的头部位置，并影响着这些位置处的存储内容。在第 5 章（以及 1.2.3.5 节）将看到，将单带图灵机推广为多带图灵机，在定义空间复杂度时至关重要。多带图灵机的另一优势是简化了函数计算机的设计。

教学注释：强烈建议不要在教学中让学生编写图灵机程序。这种练习是吃力而不讨好的。相反，学生应该有能力证明，图灵机模型与更接近实际计算机的模型在功能上完全相同（见如下的“健全性检验”），即一个函数可以由图灵机计算，当且仅当它可以后者计算。对于初学者来说，可以证明一个函数可以用单带图灵机计算当且仅当它可以用多带图灵机（如双带）计算。

丘奇—图灵命题（Church-Turing Thesis）

图灵机模型最重要的特点就是其简单性。换言之，与更“现实”的计算模型相比，图灵机的形式化描述更简单，分析起来也更简单。丘奇—图灵命题断言，采用图灵机模型并不会损失什么：一个函数可以用图灵机计算当且仅当它可以用任意其他“合理且通用”的计算模型计算。

① 图 1.2 中的机器中，在初始状态（即“a”），用符号 $\sigma+4$ 代替符号 σ ，将内部状态修改为“b”，并向右移动一格。事实上，图灵机设计时都会“标记”最左边的单元（以便于将来识别）。

② 另一种习惯是，当箭头走到最左边时停机，并定义输出为只存储比特值的带内容的最大前缀。

这个结论被称为命题而非定理，因为它针对的是一个未定义的直观概念（如“合理且通用”的计算模型）。一个计算模型是合理的，如果它只允许那些在某种直觉意义上是“简单”的计算规则。例如，我们应该可以预见能实现这些计算规则的机械装置。另一方面，计算模型应该允许计算一些直观上可以计算的“简单”函数。最后，计算模型应该能模仿图灵机（即当给定图灵机的描述以及当前配置时，通过对函数的计算能返回图灵机的后继配置）。

哲学上的解释

在任何一门科学中（或者更一般地，在任何对于直观概念进行严格推理的尝试中），命题的作用就是将直观概念与形式化定义相联系。对直观概念进行严格定义的任何一种尝试，都会导致一个形式化的定义，它不同于最初的直觉，而两者之间的关系就成为一个需要研究的问题。然而，这个问题从未被严肃对待过，因为它关系到两个事物，而其中之一是没有定义的。因此，直觉与定义之间的关系问题，永远无法用严格的方法解决（即这个问题永远属于直觉范畴）。

健全性检验：图灵机可以模仿抽象的 RAM

为了肯定丘奇—图灵命题，我们可以试着定义一个抽象的随机存取机（Random-Access Machine, RAM），并证明它可以用图灵机来模仿。一个抽象的 RAM 包含无穷多个存储单元（每个单元中能存储一个整数），类似于存储单元的、数量有限的寄存器，其中一个为程序计数器，以及由有限集中选取的指令所构成的程序。可能的指令集合包含以下指令：

- $\text{reset}(r)$: r 为寄存器序号，将寄存器 r 的值置 0。
- $\text{inc}(r)$: r 为寄存器序号，增加寄存器 r 的值。类似地， $\text{dec}(r)$ 使寄存器 r 的值减小。
- $\text{load}(r_1, r_2)$: r_1, r_2 为寄存器序号，将存储位置 m 处的值存入寄存器 r_1 ，而 m 为寄存器 r_2 中的存储内容。
- $\text{store}(r_1, r_2)$: 将寄存器 r_1 的值存入内存中，类似地有 load 指令。
- $\text{cond-goto}(r, l)$: r 为寄存器序号， l 为不大于程序长度的整数，该指令的含义是如果寄存器 r 中保存的值非负，则将程序计数器的值置为 $l-1$ 。

每条指令执行之后，程序计数器的值加 1，程序计数器的指针移到下一条将要执行的指令（当程序计数器的值超过了程序长度时停机）。可以将前 n 个存储单元的内容定义为机器的输入，其中 n 的值保存于一个特殊的寄存器中。

我们注意到这个抽象的 RAM 模型与图灵机功能一样强大（见以下的细节）。然而，为了使其更接近实际应用中的计算机，可以为这个模型增加一些额外的计算机指令，如指令 $\text{add}(r_1, r_2)$ （或 $\text{mult}(r_1, r_2)$ ）可以得到寄存器 r_1 和 r_2 的内容之和（或积），并将其存入 r_1 。建议读者自己证明，这个抽象的 RAM 可以由一个图灵机来模仿。^①（提示：注意到在模仿过程中，只需保存输入、所有寄存器的内容以及运算中访问的存储单元内容即可）。^②

① 我们强调两个模型的等价性，这是由于引入 RAM 模型的目的是为了使读者相信图灵机的计算能力并不太弱（作为一个通用计算模型而言）。然而，它们不是很强大的证据却很明显。因此，证明 RAM 模型可以模仿图灵机似乎没有什么意义。然而，注意到事实其实是这样的：使用 RAM 的内存单元，可以存储图灵机的带单元信息。

② 因此，在每次访问图灵机的带时，也同时访问了带上保存着的 RAM 的内存单元列表。当模仿一个 RAM 指令时，首先检查相关的 RAM 单元是否出现在表中，然后根据需要，或者向表中增加一项，或者修改这一项。

思考

观察到这个抽象的 RAM 模型比图灵机模型复杂得多，另外，选择一个合理的指令集（即模型允许的操作）将导致一个有缺陷的循环（因为合理的指令可能只允许“简单”操作，而后者对应着易计算的函数……）。如果允许读者只考虑实际计算机中的指令，则可以避免这个有缺陷的循环。（我们认为这种出于经验的考虑在目前情况下是合理的，因为现在的目标只是将图灵机模型与读者关于实际计算机的使用经验相结合。）

1.2.3.3 不可计算的函数

本小节的内容对本书其余章节的学习来说并不是必需的，但我们认为这里提供了一个有用的视角。

作为复杂性理论研究的专业人士，我们应该知道并非所有函数都是可计算的。事实上，这里要传递的一个重要信息是，并非每个良好定义（Well-defined）的任务都可以用一个“合理的”自动化程序解决（即程序的描述很简单，并且适用于该问题的任何一个实例）。进一步地，不可计算的函数不只是存在，而且实际上大部分函数都不可计算，只有相当少量函数才是可计算的。

定理 1.4（可计算函数的稀缺性） 可计算的函数集是可数的，而由所有函数（由串到串）构成的集合之势为 $\aleph$ 。

我们强调这个定理在任意一个合理的计算模型中都成立。其正确性只依赖于这样一个假设，即该模型下的每个机器描述起来长度都是有限的（可以用一个串来描述）。

证明：由于每个可计算的函数都可由被有限描述的机器来计算，所以在可计算的函数集与有限串集合间存在着——映射（从而，在可计算函数集与自然数集间也存在着——映射）。另一方面，在布尔函数（即把二元字符串映射为一个比特的函数）集与 $[0, 1]$ 上的实数集间也存在着——映射。该映射将每个实数 $r \in [0, 1]$ 与函数 $f: N \rightarrow \{0, 1\}$ 相关联，其中 $f(i)$ 是 r 的二进制表示中的第 i 个比特。 ■

停机问题：与 1.2.3.1 节中的讨论相反，此时还需考虑对于某些输入不会停机的机器（由这些机器计算的函数只对那些能停机的输入有定义）。这里需要再次依赖每个机器均为有限描述的假设，将机器 M 的描述记作 $\langle M \rangle \in \{0, 1\}^*$ 。停机函数 $h: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$ 定义为， $h(\langle M \rangle, x) \stackrel{\text{def}}{=} 1$ 当且仅当 M 对输入 x 停机。以下结论超越了定理 1.4，针对着一个明确的、可计算的函数。

定理 1.5（停机问题的不可判定性） 停机函数是不可计算的。

术语“不可判定”的意思是，相应的判定问题不能用一个算法解决。定理 1.5 断言，与集合 $h^{-1}(1) = \{(\langle M \rangle, x) : h(\langle M \rangle, x) = 1\}$ 相关联的判定问题不能通过算法解决（即不存在这样的算法，给定一对 $(\langle M \rangle, x)$ ，能判断 M 是否对输入 x 停机）。实际上，以下的证明表明不存在这样的算法，给定 $\langle M \rangle$ 时能判断 M 是否对输入 $\langle M \rangle$ 停机。

证明：我们要证明，即使将 h 限定到其“对角线”（Diagonal）上（即函数 $d(\langle M \rangle) \stackrel{\text{def}}{=} h(\langle M \rangle, \langle M \rangle)$ ），停机问题也仍不可解。注意到 $d(\langle M \rangle)$ 的值是指，向 M 输入关于其自身的描述时，输出的结果。当然，这样做有点“荒谬”（不过仍是合法的）。实际上，我们采取的做法可能“更糟”：用反证法，用一个假设的机器（或与该机器相关的机器）对 d 求值，

讨论 d 的值。

首先考虑一个与 d 相关的函数 d' ，并证明这个函数不可计算。在定义函数 d' 时强制规定其不可计算，即对任意机器 M ，规定 $d'(\langle M \rangle)$ 的值不同于 $M(\langle M \rangle)$ 。具体地，函数 $d': \{0,1\}^* \rightarrow \{0,1\}$ 定义为 $d'(\langle M \rangle) \stackrel{\text{def}}{=} 1$ 当且仅当 M 对于输入 $\langle M \rangle$ 停机，且输出 0（就是说，当 $d'(\langle M \rangle) = 0$ 时，或者 M 对于输入 $\langle M \rangle$ 不停机，或者输出的不是 0）。现在用反证法，假设某个机器，记作 $M_{d'}$ ，可以计算 d' 。注意到 $M_{d'}$ 对任意输入都停机，这些输入中也包括 $\langle M_{d'} \rangle$ ，从而 $M_{d'}$ 对输入 $\langle M_{d'} \rangle$ 停机。由 d' 的定义， $d'(\langle M_{d'} \rangle) = 1$ 当且仅当 $M_{d'}$ 对输入 $\langle M_{d'} \rangle$ 停机且输出 0（即当且仅当 $M_{d'}(\langle M_{d'} \rangle) = 0$ ）。因此， $M_{d'}(\langle M_{d'} \rangle) \neq d'(\langle M_{d'} \rangle)$ ，与 $M_{d'}$ 能计算 d' 的假设矛盾。

下面证明 d 是不可计算的，从而 h 不可计算（因为对任意 z ， $d(z) = h(z, z)$ ）。为了证明 d 不可计算，我们证明如果 d 可计算，则 d' 也可计算（已经证明 d' 不可计算，从而可以由逆否命题得出 d 不可计算）。仍用反证法，假设 A 为计算 d 的算法（即对任意机器 M ，有 $A(\langle M \rangle) = d(\langle M \rangle)$ ）。下面构造一个计算 d' 的算法：给定输入 $\langle M' \rangle$ ，对于输入 $\langle M' \rangle$ 调用 A ，其中 M'' 以如下方式运行：

- (1) 对于输入 x ， M'' 模仿 M' 对 x 的操作。
- (2) 如果 M' 对输入 x 停机且输出 0，则 M'' 也停机。
- (3) 如果 M' 对输入 x 停机且输出不为 0，则 M'' 进入无限循环（不停机）。
- (4) 否则（即 M' 对输入 x 不停机）， M'' 不停机（因为它会永远停留在第一步）。

注意到由 $\langle M' \rangle$ 到 $\langle M'' \rangle$ 的映射是易于计算的（通过向 M' 增加检查输出的指令以及需要时进入无限循环的指令即可），且有 $d(\langle M'' \rangle) = d'(\langle M' \rangle)$ 成立，因为 M'' 对输入 x 停机当且仅当 M' （译者注：原著中误为 M'' ）对 x 停机且输出 0。由此我们得到了一个计算 d' 的算法（即将 $\langle M' \rangle$ 的输入作为 $\langle M'' \rangle$ 的输入，并输出 $A(\langle M'' \rangle)$ ），这与前面已经证明的 d' 不可计算的结论矛盾。 ■

图灵归约

定理 1.5 证明的第二部分的核心是，构造解一个问题的算法（即求解 d' 的算法），其中将解另一个问题的算法（即求解 d （或 h ））用作子程序。实际上，第一个算法更像是一个针对“具体功能”的算法体系，而非一个实际上（需要实现）的子程序，并且这个算法也许并不存在。这样的算法体系称为图灵归约（Turing-reduction，其形式化定义见 1.2.3.6 节）。因此，我们将对 d' 的计算图灵归约为对 d 的计算，又将对 d 的计算图灵归约为对 h 的计算。由函数 f' 到 f 的图灵归约“自然的”（“积极的”）含义是：当给定一个计算 f 的算法时，可以得到计算 f' 的算法。然而，在定理 1.5 的证明中，采用了“不自然的”（“消极的”）反向方法：如果（已知）不存在计算 $f' = d'$ 的算法，则也不存在计算 $f = d$ 的算法（要证明的结论）。后面我们将介绍资源受限的图灵归约（即多项式时间归约），它在复杂性理论中起着核心作用，而且它们大多数时候是以“反向”的方法使用的。我们将定义这种归约并在后续章节中大量使用。

莱斯定理（Rice's Theorem）

停机问题的不可判定性（或函数 d 的不可计算性）是另一个更普遍现象的特殊情况，这个普遍现象是：如果一个非平凡的判定问题与由一个给定图灵机可计算的函数相关，则该判定问题不存在求解算法。以下详述这个定理，并澄清前面对各个问题类的定义。（我们将再次

使用对所有输入都不停机的图灵机。)

定理 1.6 (莱斯定理) 设 $\mathcal{F}$ 为所有可计算的部分函数构成集合的任意一个非平凡子集,^① 令 $S_{\mathcal{F}}$ 为字符串集合, 这些串描述了能计算 $\mathcal{F}$ 中函数的所有机器, 则 $S_{\mathcal{F}}$ 中成员的判定问题没有求解算法。

可以构造从 d 向 $S_{\mathcal{F}}$ 中成员的判定问题的图灵归约来证明定理 1.6。这里不给出具体证明过程, 因为这与本书的主旨关系不大。我们强调定理 1.5 和定理 1.6 在任意合理的计算模型中都成立 (既指潜在的求解者, 也指被求解者当作输入的机器描述)。因此, 定理 1.6 的含义是: 不存在这样的算法, 可以判定由给定计算机程序 (用任意编程语言写成) 计算的函数的任意一种非平凡性质。例如, 不存在这样的算法, 可以判定是否一个给定的计算机程序对每个可能的输入都停机。显然, 这个断言对于程序验证具有重大意义。

波斯特对应问题 (Post-Correspondence Problem)

我们认为, 在计算设备 (作为输入) 之外的领域中, 也存在着不可判定性。具体地, 考虑如下的波斯特对应问题, 其输入是两个字符序列, $(\alpha_1, \alpha_2, \dots, \alpha_k)$ 和 $(\beta_1, \beta_2, \dots, \beta_k)$, 问是否存在一个下标序列 $i_1, i_2, \dots, i_l \in \{1, 2, \dots, k\}$, 使得 $\alpha_{i_1} \dots \alpha_{i_l} = \beta_{i_1} \dots \beta_{i_l}$ 。(我们强调序列的长度是有限的。)^②

定理 1.7 波斯特对应问题是不可判定的。

证明过程仍省略, 其中涉及到从 d (或 h) 到该问题的图灵归约。^③

1.2.3.4 通用算法

迄今为止, 我们使用了这样一个假设, 即在任意合理的计算模型下, 每个机器 (或计算规则) 的描述是有限的。另外, 还用到一个事实, 即计算模型应该允许修改机器的描述, 以使得到的新机器能计算一个明显相关的函数 (见定理 1.5 的证明)。在这里更进一步地, 假设机器的描述 (在这个模型中) 在以下意义上是“有效”的: 存在一个算法, 给定某个机器 (或计算规则) 的描述以及相应的环境说明, 算法能够通过在该环境下执行机器的一个步骤而确定结果 (或一次应用计算规则的效果)。该算法反过来可以运行于上述计算模型中 (假设模型是通用的, 见丘奇—图灵命题)。连续应用该算法, 便得到通用机的概念, 下面我们用工图灵机术语来形式化地 (具体) 定义通用机。

定义 1.8 (通用机) 通用图灵机是指这样一类图灵机: 输入机器 M 的描述及 x , 若 M 对 x 停机, 则返回 $M(x)$, 否则不停机。

也就是说, 一个通用图灵机可以计算定义于 $(\langle M \rangle, x)$ 上的部分函数 u , 使得 M 对于输入 x 停机, 此时有 $u(\langle M \rangle, x) = M(x)$ 成立。即 $u(\langle M \rangle, x) = M(x)$ 当且仅当 M 对于输入 x 停机, 否则 u 在 $(\langle M \rangle, x)$ 上无定义。注意: 如果 M 对于所有可能的输入都停机, 则 $u(\langle M \rangle, x)$ 对所有的 x 都有定义。

在此强调, 给出一个事物 (如通用机) 的定义并不代表它一定存在。然而, 在前面的讨论中已经暗示了通用图灵机的存在性, 这对那些写过计算机程序 (并深入思考过计算机编程)

① 集合 S 称为 U 的非平凡子集, 如果 S 及 $U \setminus S$ 均为非空集。显然, 如果 $\mathcal{F}$ 为可计算函数的平凡子集, 则相应的判定问题可以由一个“平凡”的算法解决, 这个算法只输出相应的固定比特。

② 相反, 是否存在足够多具有特定长度的序列的问题, 可以在这个长度的指数时间内判定。

③ 我们指出归约将 h 的实例 $(\langle M \rangle, x)$ 映射为序列对 $((\alpha_1, \alpha_2, \dots, \alpha_k), (\beta_1, \beta_2, \dots, \beta_k))$, 使得只有 α_i 和 β_i 由 x 确定, 而 k 及其他串只依赖于 M 。

的人来说是显而易见的。

定理 1.9 通用图灵机是存在的。

定理 1.9 断言，部分函数 u 是可计算的。相反，可以证明将 u 以任意方式扩展为完整函数之后则不可计算。即假设存在任意的完整函数 $\hat{u}$ ，在 u 有定义的输入上与 $\hat{u}$ 是一致的，则 $\hat{u}$ 不可计算。^①

证明：给定一个对 $(\langle M \rangle, x)$ ，只需模拟机器 M 对输入 x 的计算。这个模拟过程很直接，因为（由 M 的效果）可以用迭代方法确定 M 对输入 x 进行计算时，下一个时刻的配置。如果该计算停机，则可以得到其输出（于是，对于输入 $(\langle M \rangle, x)$ ，算法返回 $M(x)$ ），否则，模拟过程将成为无限的计算过程，这意味着算法对于输入 $(\langle M \rangle, x)$ 不停机。因此，上述模拟过程就构成了一个通用机（即得到了计算 u 的算法）。■

前面暗示过，通用机器的存在性这一基本事实正是通用计算机的基础。事实上，一个专用图灵机（或算法）是解决某个特定问题的设备。先验地，解每个问题时，需要构造一个新的物理设备，使该问题可以在物理世界中（而非想象中的实验）被解决。通用机的存在性表明通用计算机这种物理设备是可以构造出来的，从而，要解任意一个特定问题时，只需编写相应的程序并在通用计算机上执行（或模仿）即可。因此，通用机对应于通用计算机，并提供了将硬件与软件相分离的基础。换言之，通用机的存在性说明了软件可以（部分地）被视为输入。

除了实用价值，通用机（及其变形）的存在性对于可计算性理论和计算复杂性理论也有着重要意义。为证实这一点，注意到定理 1.6 中蕴含了对于某些输入带而言，某个固定通用机的行为是不可判定的。例如，由此可以断定，对某些固定的机器（即通用机），不存在算法可以判断该机器是否能对给定输入停机。回顾定理 1.7 的证明，可以得到结论，即使将输入序列限制为固定长度（即 k 为固定值）时，Post 对应问题也是不可解的。以下给出通用机在可计算性理论中的一个重要应用。

绕个弯子：柯尔莫果洛夫复杂度（Kolmogorov Complexity）

通用机的存在性可以看作是描述物体的一种有效且清晰的通用语言，它在柯尔莫果洛夫复杂度中起核心作用。不严格地说，后者重点考虑的是物体的（有效）描述所需的长度，并将最小长度当作该物体的固有“复杂度”，就是说，“简单”物体（或现象）的描述（或解释）也比较短，而“复杂”物体绝不会有有一个短的描述。当然，这些（有效）描述必须参考某种固定的“语言”（即对于一个固定的机器而言，给定一个物体的简要描述，该机器会生成其确切描述）。对任意机器 M ，如果 $M(x) = s$ ，则串 x 称为 s 关于 M 的描述。 s 关于 M 的最短描述称为 s 关于 M 的柯尔莫果洛夫复杂度，记作 $K_M(s)$ 。显然，我们希望固定 M ，从而使每个串都有关于 M 的描述，另外，还要使这种描述不会“明显地”比关于另一个机器 M' 的描述更长。以下定理很自然地将通用机当作“参考点”（即上述的 M ）。

定理 1.10（通用机的复杂度） 设 U 为通用机，则对任意机器 M' ，存在一个常数 c 使

① 这个断言是易证明的，完整的函数 $\hat{u}$ 以那些在 u 中无定义的符号作为输入时，将输出一个特殊符号 $\perp$ （即如果 u 在 $(\langle M \rangle, x)$ 上无定义，则 $\hat{u}(\langle M \rangle, x) \stackrel{\text{def}}{=} \perp$ ，否则， $\hat{u}(\langle M \rangle, x) \stackrel{\text{def}}{=} u(\langle M \rangle, x)$ ）。此时， $h(\langle M \rangle, x) = 1$ 当且仅当 $\hat{u}(\langle M \rangle, x) \neq \perp$ ，故停机函数 h 可以图灵归约到 $\hat{u}$ 。一般在定理 1.5 的证明中利用这样一个事实，即当机器 M 对所有输入都停机时，则对任意 x ，有 $\hat{u}(\langle M \rangle, x) = u(\langle M \rangle, x)$ （特别地，当 $x = \langle M \rangle$ 时等式也成立）。

得对任意串 s , 有 $K_U(s) \leq K_{M'}(s) + c$ 。

由上述定理可以得出, (如果令 $c = O(|\langle M' \rangle|)$ 并) 观察到如果 x 是 s 关于 M' 的一个描述, 则 $(\langle M' \rangle, x)$ 是 s 关于 U 的描述。这里的关键是要对一对串进行充分的编码 (如将对 $(\sigma_1 \cdots \sigma_k, \tau_1 \cdots \tau_l)$ 编码为 $\sigma_1 \sigma_1 \cdots \sigma_k \sigma_k 01 \tau_1 \cdots \tau_l$)。给定任意一个通用机器 U , 串 s 的柯尔莫果洛夫复杂度定义为 $K(s) \stackrel{\text{def}}{=} K_U(s)$ 。易验证以下事实成立:

- (1) 对任意 s , $K(s) \leq |s| + O(1)$ 。(提示: 对于计算同一映射的机器应用定理 1.10。)
- (2) 存在无穷多个串 s 满足 $K(s) \ll |s|$ 。(提示: 考虑 $s = 1^n$ 的情况, 或者可以考虑对所有 x , 满足 $|M(x)| \gg |x|$ 的任意机器 M 。)
- (3) 某些长为 n 的串具有至少为 n 的复杂度。进一步地, 对任意 n 和 i , 有

$$\left| \left\{ s \in \{0,1\}^n : K(s) \leq n-i \right\} \right| < 2^{n-i+1}$$

(提示: 不同的串关于机器 U 有不同的描述)

可以证明, 函数 K 是不可计算的。证明关系到自然数的下述“描述”产生的矛盾: 最大的自然数可以用由 1000 个字母组成的英文句子来描述 (这意味着如果上述最大自然数的定义是良好的, 则这个最大数字的后继也可以用由 1000 个字母组成的英文句子来描述)。当然, 这种说法预先假设了任意英文句子在某种意义下 (如在柯尔莫果洛夫复杂度含义下) 都是合法的描述。具体地, 这种说法假设我们可以确定每个自然数的柯尔莫果洛夫复杂度, 并且进一步地假设, 可以高效地生成在某个阈值范围内具有最大柯尔莫果洛夫复杂度的最大整数。事实上, 这个矛盾证明了这样一个事实, 即后一个任务是不可能实现的, 也就是说, 不存在一个算法, 在给定 t 时, 可以生成以词典顺序排列的最后一个满足 $K(s) \leq t$ 的串 s , 因为如果这样的算法存在, 设其为 A , 则 $K(s) \leq O(|\langle A \rangle|) + \log t$ 且 $K(s0) < K(s) + O(1) < t$, 从而与 s 的定义矛盾。

1.2.3.5 时间及空间复杂度

给定一种计算模型 (如图灵机), 考虑对每个输入都停机的算法。算法对每个可能输入执行的步骤数量 (即应用计算规则的次数), 称为算法 (或机器) 的时间复杂度 (Time Complexity), 就是说, $t_A: \{0,1\}^* \rightarrow \mathbb{N}$ 称为算法 A 的时间复杂度, 如果对任意输入 x , A 恰在 $t_A(x)$ 步后到达停机状态。

我们最感兴趣的是时间复杂度与具有最大长度的输入的关系。即对于上述的 t_A , 考虑函数 $T_A: \mathbb{N} \rightarrow \mathbb{N}$, $T_A(n) \stackrel{\text{def}}{=} \max_{x \in \{0,1\}^n} \{t_A(x)\}$ 。有时也用 T_A 表示 A 的时间复杂度。

问题的时间复杂度

在前言及概述中提到, 复杂性理论通常并不考虑某个特定算法的 (时间) 复杂度, 而是考虑问题的 (时间) 复杂度, 假设该问题可解 (由某个算法)。直观上, 问题的时间复杂度定义为解该问题的最快算法的时间复杂度 (假设“最快算法”有定义)^①。实际上, 人们也许会对解某个问题的算法 (时间) 复杂度的上限和下限感兴趣。因此, 当我们说某个问题 Π 的复

① 4.2.2 节将指出 (见定理 4.8), 假设解某个问题总是存在“最快算法”并不总是合理的。另一方面, 这个假设在某些重要情况 (见定理 2.33) 下则是合理的。但是, 即使在这些情况下, 该算法也只是关于一个常数因子是“最快的” (或“最优的”)。

杂度为 T 时, 其真正含义是 Π 的复杂度至多为 T 。类似地, 如果说问题 Π 需要的时间为 T , 则意思是 Π 的时间复杂度至少是 T 。

回顾前面对于某个固定计算模型的讨论。事实上, 问题 Π 的复杂度也许依赖于实现解决问题的算法所需的特定计算模型。以下的 Cobham-Edmonds 定理表明, (时间复杂度的) 这种变化不会太大, 特别地, 与复杂性理论目前所关注的焦点 (如 P-vs-NP 问题) 无关。

Cobham-Edmonds 定理

刚才说过, 问题的时间复杂度可能取决于计算模型。例如, 集合 $\{xx : x \in \{0,1\}^*\}$ 的成员判定问题可以利用一个双带图灵机在线性时间内解决, 但是在单带图灵机上却需要平方时间。^① 另一方面, 在多带图灵机模型中时间复杂度为 t 的问题, 在单带图灵机模型中的复杂度为 $O(t^2)$ 。Cobham-Edmonds 定理表明, 问题在任意两个“合理且通用”的计算模型中的时间复杂度是多项式相关的。即一个问题在某个“合理且通用”的计算模型中时间复杂度为 t , 当且仅当其在 (单带) 图灵机模型中的时间复杂度为 $\text{poly}(t)$ 。

事实上, Cobham-Edmonds 定理强化了丘奇-图灵定理。它不仅表明, 只要计算模型是“合理且通用”的, 则可解的问题类保持不变, 而且指出在这些模型中 (可解问题) 的时间复杂度是多项式相关的。

高效算法

上述讨论中暗示, 复杂性理论在很大程度上与高效算法相关。后者被定义为多项式时间算法 (即算法的时间复杂度上限为其输入长度的多项式)。但是由 Cobham-Edmonds 定理, 这类算法的定义与所选择的“合理且通用”的计算模型无关。高效算法与多项式时间可计算性的关系建立于以下两个考虑之上:

- 哲学上的考虑: 直观上, 高效算法是指那些可以在有限步骤内实现的算法, 其执行步骤数随输入长度的增长而缓慢增加。为了读取整个输入, 至少需要线性的时间复杂度。另一方面, 要避免使用明显运行较慢的算法, 特别是“穷尽搜索”算法, 这些算法需要指数时间。另外, 对于高效算法类的良好定义应该在算法的自然合成下保持封闭性 (还应关于合理计算模型以及问题实例编码方式的微小改变保持健壮性)。

选择多项式作为高效算法的时间界限可以满足上述所有要求: 多项式构成了一类缓慢增长函数的“封闭”集, 其中“封闭”的意思是在加法、乘法及函数合成运算下的封闭性。这些封闭性条件保证了高效算法类在算法的自然合成运算下的封闭性 (以及在任意合理且通用计算模型下的健壮性)。另外, 多项式时间算法可以执行显然是很简单的计算 (虽然不一定是平凡的计算), 另一方面, 这类算法不包括效率很低的算法 (如穷尽搜索)。

- 经验上的考虑: 显然, 实用中被认为是高效的算法其运行时间局限于一个小的多项式内 (至少是关于实际输入的多项式)。问题在于是否任意多项式时间算法在直观上都是高效的。根据以往经验, 人们认为每个可在多项式时间内解决的自然问题也都存在一个“合理且高效”的算法。

① 假设后一个事实是非平凡的, 则根据由通信复杂度问题^[148], (12.2 节) 出发的“归约”, 可得到一种证明。直观上, 判定上述集合成员的单带图灵机可以被视为输入的两方之间的通信信道。主要讨论形如 $y0^n z0^n$, $y, z \in \{0,1\}^n$ 的输入, 机器从第一方向第二方的每次移动都 (在其内部状态中) 携带了 $O(1)$ 比特信息, 并至少运行 n 步。调用 (两方) 识别函数 (即当 $y=z$ 时, $\text{id}(y,z)=1$, 否则 $\text{id}(y,z)=0$, 见参考文献[148] (第 1 章)) 通信复杂度的线性下界, 即可完成证明。

我们强调,就复杂性理论中的大部分概念、结论及问题而言,高效算法与多项式时间计算之间的联系并不是必需的。从任何一个满足上述封闭性条件,且允许实现某些简单而不十分复杂的计算的算法类出发,都可以得到类似的结论,虽然形式化地描述这些结论可能更加复杂。特别地,本书涉及到的所有结论及问题都与不同计算任务复杂度之间的关系有关(而不是与某些计算任务复杂度的精确取值相关)。通过将一个问题在时间复杂度上界转化为另一个问题的时间复杂度上界,可以清晰地表明这种关系^①。这种冗长的阐述将保留标准阐述的内容,只是更加复杂。因此,我们遵循以讨论多项式时间计算为主的习惯,并强调这样做不仅是自然的,而且提供了最简单的方法来处理高效计算中的基本问题。

再谈通用机

从时间复杂度的概念出发,可以得到通用函数 u (见 1.2.3.4 节) 的时间受限版本。具体地,如果对于输入 x , 机器 M 在 t 步内停机并输出串 y , 则定义 $u'(\langle M \rangle, x, t) \stackrel{\text{def}}{=} y$, 而当机器 M 对输入 x 的处理超过了 t 步时, 定义 $u'(\langle M \rangle, x, t) \stackrel{\text{def}}{=} \perp$ 。函数 u' 则与 u 不同, 它是一个完整函数。另外, 也与任何由扩展 u 而得到的完整函数不同, u' 是可计算的。最后, u' 可由机器 U' 计算, 对于输入 $X = (\langle M \rangle, x, t)$, U' 在 $\text{poly}(|X|)$ 步之后停机。事实上, U' 是通用机器的一种变型(即对于输入 X , U' 只模仿 M 在 t 步的运行, 而不是一直模仿 M 直到其停机(有可能 M 永不停机))。注意到 U' 的运行步骤数取决于具体计算模型(这里不可避免地会产生某种负载, 因为对 M 的每一步模仿都需要先读取 M 描述中的相应部分)。

空间复杂度

另一个衡量算法(或计算任务)“复杂程度”的自然量度是计算过程中使用的存储空间。这里是指用于存储计算的中间结果的存储空间。由于我们讨论的重点是那些使用存储空间为输入的亚线性函数的算法, 因此十分有必要使用一个模型, 可以区分用于保存原始输入和最终输出的存储空间以及用于保存计算中间结果的存储空间。在图灵机模型中, 可以通过多带图灵机来实现这一点, 在多带图灵机中, 用一个专用的只读带来保存输入(称为输入带(Input Tape)), 用一个专用的只写带来保存输出(称为输出带(Output Tape)), 而中间结果保存于工作带(Work-tape)上。因此, 输入和输出带是不能用于保存中间结果的。这样一个机器 M 的空间复杂度定义为函数 s_M , 对于输入 x , $s_M(x)$ 表示 M 在计算中扫描的工作带的长度。类似于时间复杂度, 我们通常定义 $S_A(n) \stackrel{\text{def}}{=} \max_{x \in \{0,1\}^n} \{s_A(x)\}$ 。

1.2.3.6 预言机

1.2.3.3 节中讨论的图灵归约可以用如下定义的预言机(Oracle Machine)来解释。不严格地说, 预言机是一种强化的机器, 它可以向外界提问题。我们考虑这样一种情况, 这些被称为询问(Query)的问题, 始终由某个函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 来回答, 该函数称为预言(Oracle)。也就是说, 如果机器发出询问 q , 则可以得到应答 $f(q)$ 。在这种情况下, 称预言机可以访问预言 f 。对于预言机 M , 串 x 和函数 f , 用 $M^f(x)$ 表示 M 能访问预言 f 时, 对于输入 x 的输出

^① 例如, SAT 问题的 NP-完全性(见定理 2.22)表明, 从任意一个可在时间 T 内解 SAT 问题的算法出发, 可以得到在时间 T' 内分解合数的算法, 其中 $T'(n) = \text{poly}(n) \cdot (1 + T(\text{poly}(n)))$ 。(更一般地, 对于某个搜索问题 R 的 n 比特实例而言, 如果其解的正确性可在时间 $t(n)$ 内验证, 则有关 SAT 的假设暗示了, 这个(R 的 n 比特实例的)解可在时间 T' 内求得, 其中 $T'(n) = t(n) \cdot (1 + T(O(t(n))^2))$ 。

(重新审视一定理 1.5 证明中的第二部分, 实际上描述了一个预言机, 当访问预言 d 时, 能计算 d')。

预言机是对标准计算设备(机器)概念的一种扩展, 其严格定义则是后者的形式化模型的扩展。特别地, 通过对图灵机模型的拓展, 我们可以得到以下的预言图灵机模型。

定义 1.11 (预言的使用)

- 预言机是一种特殊的图灵机, 其中增加了一个附加带及两个特殊状态, 这个附加带称为预言带, 而两个特殊状态为预言调用 (Oracle Invocation) 及预言对话 (Oracle Spoke)。

- 预言机 M 在能访问预言函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 时, 对输入 x 的计算是建立在后继配置函数之上的。在非预言调用状态, 后继配置的定义与图灵机相同。设 γ 为预言请求状态下机器的配置, 并假设预言带中存储内容为 q (即 q 为保存比特值的带上的最大前缀), ^① 则 γ 的后续配置与 γ 基本相同, 只是状态变为预言通话, 而预言带的存储内容变为 $f(q)$ 。串 q 称为 M 的请求, $f(q)$ 称为预言的应答。

- M 在可以访问 f 时对输入 x 的输出记作 $M^f(x)$ 。

我们强调预言机的运行时间为预言机(自身)在计算中运行的步数, 而预言对每个请求的应答可在一步之内得出。

1.2.3.7 受限的模型

在可计算性理论中常遇到受限的计算模型, 但是它们与本书无关。这种模型包括有限自动机模型, 它与空间复杂度为零(或者等价地, 为一个常数)的图灵机在某种程度上是一致的。

我们认为, 研究这种受限计算模型最重要的动机是它们为某些自然(或人为)现象提供了简单模型。然而, 这个动机似乎与对各种计算任务复杂性的研究相距甚远, 后者考虑的是通用计算模型, 并求在这些模型中计算的复杂度。

教学注释: 事实上, 我们摒弃了通常将可计算性理论与形式语言和自动机理论相提并论的做法。虽然在历史上, 这两种理论间的联系(至少在西方)无可否认, 但这个事实并不足以表明, 将两种本质上完全不同的理论相提并论是合理的(特别是这种联系会导致对于可计算性理论的错误认识时)。因此, 我们的观点是, 对 Chomsky 层级^[123](第 9 章)中任意较低层次的研究应该与计算理论(更不用说复杂性理论)的研究相分离。

1.2.4 非一致性计算模型(电路及建议)

卡蜜儿: 就像塞尔玛和路易丝^②。但是……没有枪。

派翠: 哦, 好吧, 没有枪, 我不知道……

帕特丽夏·罗兹玛, 夜幕低垂, 1995

本书介绍非一致性计算模型的主要目的, 是将其作为某些计算问题的根源(见 2.3.3.1 节及定理 5.4)。另外, 在 3.1 节和 4.1 节也简要介绍了这些模型。

① 这与图灵机中带上存储的初始内容的定义(见 1.2.3.2 节)相符。一个普遍做法是仅当机器头部停留在预言带的最左边时才激活预言。我们指出, 在空间复杂性语义下需要使用两个预言带: 一个只写带, 用于查询; 一个只读带, 用于应答。

② 电影《末路狂花》主角, 译者注。

非一致性计算模型的含义是,对每种可能的输入长度都要考虑不同的计算设备,而不存在与“一致性”要求相关的设备能够对不同的输入长度做出响应。更进一步,这些设备的集合在本质上是一个无限集,并且(当不存在一致性要求时)甚至不存在一种有限的描述。尽管如此,集合中每个设备的描述仍是有限的。事实上,设备长度(或其描述的长度)与输入长度之间的关系将成为要重点考虑的问题。

研究非一致性计算模型的目的,既包括下限技术的开发,也有高效算法在功能上的简化。^①

这两种情况都要求去掉一致性要求,其目的一是为了简单化,二是希望(并相信)这样做并没有在本质上丢失任何东西。在开发下限时,人们希望对所有参数的限制(即输入长度及设备描述的有限性)可以允许利用组合技术来分析某种参数环境下的局限性。

我们主要讨论两种与非一致计算设备相关的模型:布尔电路(Boolean Circuits,见1.2.4.1节)及“接受建议的机器”(Machines That Take Advice,1.2.4.2节)。前者可以用于研究计算的演化(即下限技术的开发),而后者更适合作为模型来使用(如对高效算法的功能做出限制)。

1.2.4.1 布尔电路

最为人熟知的非一致性计算模型是布尔电路。历史上这种模型是为了描述实际电路的“逻辑操作”而引入的。然而,戏剧性地,今天该模型为复杂性理论中某些与实用最无关的研究(其目的是开发一些方法来帮助人们理解高效算法的内在局限性)提供了平台。

布尔电路简单地说是有一个有向无圈图,^②图中对顶点进行了标记。为了简化讨论,我们假设图中无独立顶点(即没有输入边和输出边的顶点),从而图中的顶点可分为三种类型:源顶点、目的顶点和内部顶点。

(1) 内部顶点是既有输入边也有输出边的顶点(即出度和入度至少为1)。在布尔电路中,内部顶点称为门(Gates)。每个门标记为一种布尔操作,典型操作有 $\wedge$ 、 $\vee$ 和 $\neg$ (对应着“与”“或”和“非”)。另外,还要求标记为“ $\neg$ ”的门入度为1。 $\wedge$ -门和 $\vee$ -门的入度可以是任意大于0的整数,对于出度也有同样的要求。

(2) 图中的源顶点(即无输入边的顶点)称为输入端(Input Terminals)。每个输入端用一个自然数标记(这个数被看作是输入变量的索引)。(为了定义表达式(见1.2.4.3节),允许不同的输入端可以用同一个数标记。)^③

(3) 图中的目的顶点(即无输出边的顶点)称为输出端(Output Terminals),要求其入度为1。每个输出端也用一个自然数标记,如果电路有 m 个输出端,则它们被标记为 $1, 2, \dots, m$ 。从而,不允许不同的输出端用同一个数标记,并强调所有输出端必须用连续的自然数来标记(事实上,输出端的标记对应于电路输出位置的索引)。

为简单起见,我们强制性地要求输入端也用连续的自然数标记。^④

有 n 个输入和 m 个输出的布尔电路诱导(实际上可计算)出一个 $\{0,1\}^n$ 到 $\{0,1\}^m$ 的函数,该函数定义如下:对任意给定串 $x \in \{0,1\}^n$,迭代地为电路中的顶点赋值,使得为输入端分配 $x = x_1 x_2 \dots x_n$ 中的相应比特,并以自然方式确定其他顶点的值。

① 后一种情况主要针对着这样的高效算法:给定一对输入(具有(多项式)相关的长度),使得研究这种算法时,可以固定(任意)一个输入,并(随机地)改变另一个输入。典型例子包括去随机化(见8.3节)和零知识证明(见9.2节)。

② 见附录G1。

③ 在一般的电路中不会出现这种情况,因为只需令同一输入端的输出边对应着多个门即可避免。注意:虽然如此,布尔表达式是不允许这样的,因为布尔表达式要求所有的非目的顶点出度恰为1。

④ 如果要构造忽略某些输入值的电路,则这个习惯使构造过程稍微有些复杂化了。具体地,我们使用输入边来自于相应输入端的人造部分,并计算出一个充分的常数。为避免将这个常数作为输出端,我们将其送入一个辅助门,而后的值由其他输入边确定(如将常数0送入 $\vee$ 门),见图1.3中对 x_3 的处理。

- 对标记为 $i \in \{1, 2, \dots, n\}$ 的输入端分配 x 的第 i 个比特 (即 x_i)。
- 如果某个 (入度为 d 的) 标记为 $\wedge$ 的门的孩子顶点被赋值为 $v_1, v_2, \dots, v_d$, 则为该门分配的值是 $\bigwedge_{i=1}^d v_i$ 。类似地, 可以确定 $\vee$ 门 (及 $\neg$ 门) 的值。

实际上, 电路是无圈图这一假设暗示了如下确定电路中顶点值的过程是良好定义的: 只要某些顶点的值未定, 则存在一个值未定的顶点, 其孩子结点的值是已经确定的, 从而, 这个过程可以继续下去, 最后能确定所有顶点 (包括输出端) 的值。

电路对于输入 x 的取值 (即输入为 x 时电路的输出) 为 $y = y_1 y_2 \dots y_m$, 其中 y_i 是根据上述过程为标记为 i 的输出端分配的值。注意: 对于给定的电路 C 和输入 x , 存在一个多项式时间算法, 能计算出 C 对于 x 的取值。这个算法由输入端到输出端依次确定电路中顶点的取值。

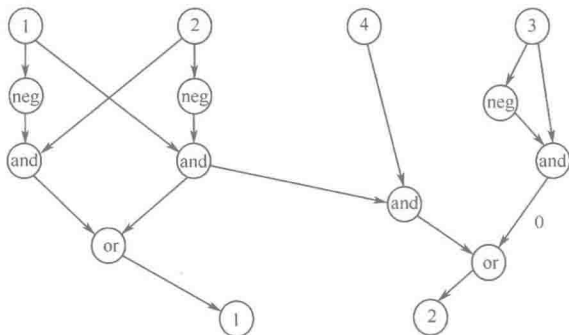


图 1.3 用电路计算函数 $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2, x_1 \wedge \neg x_2 \wedge x_4)$

设 $(C_n)_{n \in \mathbb{N}}$ 为电路簇, 函数 $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$, 如果对任意的 n , 电路 C_n 能计算将输入限定为长为 n 的串时 f 的值, 则称电路簇 $(C_n)_{n \in \mathbb{N}}$ 计算 f 。换言之, 对任意 $x \in \{0, 1\}^*$, 必有 $C_{|x|}(x) = f(x)$ 。

有界扇入与无界扇入

我们感兴趣的是这样一种电路, 其中每个门最多有两个输入边。此时允许的 (二元) 布尔操作的类型并不重要 (只要是“完整基底”的操作, 即这组操作可以用其他任意的二元布尔运算来实现)。这种电路称为有界扇入电路。相反, 其他一些研究工作考虑的是无界扇入电路, 其中每个门可能有任意多个输入边。显然, 在无界扇入电路中, 对允许布尔操作集的选择很重要, 并且只能考虑那些“一致”的操作 (适用于所有操作数, 如 $\vee$ 和 $\wedge$)。

作为复杂性量度的电路规模

电路的规模是指其边的个数。对于计算函数 $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 的电路簇 $(C_n)_{n \in \mathbb{N}}$, 可以将 C_n 的规模看作是函数 s 关于 n 的函数。具体来说, 如果对任意 n , C_n 的规模为 $s(n)$, 则该电路簇的规模复杂度为 $s: \mathbb{N} \rightarrow \mathbb{N}$ 。函数 f 的电路复杂度, 记作 s_f , 是所有可计算 f 的电路簇的规模复杂度的下确界。换言之, 对每个 n , 可以考虑将 f 限制到 n 比特串时 (记作 f_n) 计算 f 的最小电路规模, 并相应地设置 $s_f(n)$ 。我们强调非一致性在这个定义中并无明显体现, 因为对于计算具有不同长度输入的函数的各种电路之间的关系, 并未给出明确的条件。^①

① 前沿性注释: 还要注意的, 电路模型以及复杂性量度 (电路规模) 支持优化的计算设备: 每个函数 f 具有唯一的规模复杂度 s_f (而不仅仅规定其复杂度的上下界)。

函数的电路复杂度

在这里我们强调关于函数电路复杂度的一些简单事实（这些事实显然与 1.2.3.4 节中提到的柯尔莫果洛夫复杂度有关）。

(1) 最重要的一点是，任意一个布尔函数都可由某类电路计算，所以函数的电路复杂度是一个良好定义。进一步地，函数的电路复杂度最多为指数级。

（提示：函数 $f_n: \{0,1\}^n \rightarrow \{0,1\}$ 可由规模为 $O(n2^n)$ 的电路计算，该电路实现的是查表运算。）

(2) 一些函数具有多项式的电路复杂度。特别地，任意一个时间复杂度为 t 的函数（即可由时间复杂度为 t 的算法计算的函数）具有 $\text{poly}(t)$ 的电路复杂度。另外，与函数相关的电路簇是一致的（以下将在通常意义下讨论这个问题）。

（提示：考虑计算该函数的图灵机，及其对于 n 比特输入的计算。相关计算过程可以用一个 $t(n)$ 层电路模仿，电路的每一层代表机器的一种中间配置，相继配置间的关系可以解释为电路中（“一致”）的本地部分。定理 2.21 的证明中给出了更多的细节，其中构造了一个类似的模仿过程。）

(3) 几乎所有的布尔函数都有指数级的电路复杂度。特别地，可由某个规模为 s 的电路计算的 $\{0,1\}^n$ 到 $\{0,1\}$ 的映射个数小于 s^{2^s} 。

（提示：有 v 个顶点和 s 条边的电路数量最多为 $\left(2 \cdot \binom{v}{2} + v\right)^s$ 。）

注意到第一个事实暗示了电路簇可以计算那些无法用算法计算的函数。另外，这种现象在只考虑那些具有多项式规模的电路簇时也会发生，见 1.2.4.2 节中的进一步讨论。

一致性类

多项式规模的电路簇 $(C_n)_{n \in \mathbb{N}}$ 被称为是一致的，如果给定 n ，可以在 $\text{poly}(n)$ 时间内构造出电路 C_n 。注意到如果一个函数可以由一致的多项式规模电路簇计算，则它也可以由多项式时间算法计算。这个算法首先构造出一个充分的电路（由一致性假设，这个过程可在多项式时间内完成），然后对给定输入计算电路的输出（所需时间为电路规模的多项式）。

注意：对任意多项式规模电路簇计算能力的限制在（多项式规模的）一致性类中肯定成立，而这又限制了多项式时间算法的计算能力。因此，从函数电路复杂度的下限中可以得出类似的关于其时间复杂度的下限。进一步地，如同数学和自然科学中的常见情形，去掉并未被深入理解的辅助条件（如一致性）可以得到很好的结果。事实上，在研究受限制的电路类时，情况确实如此，见 1.2.4.3 节（及附录 B.2）。

1.2.4.2 接受建议的机器

对于计算设备的“非一致性程度”而言，一般的（非一致的）电路类和一致的电路类是两个极端。直观上讲，对于前者，非一致性只受设备规模的限制，而后的非一致性恒为 0。这里我们考虑一种计算模型，能根据非一致性程度来区分计算设备的规模，其中非一致性的取值范围是从 0 到设备的规模。特别地，我们考虑“接受非一致建议”的算法，它只依赖于输入长度。此时的非一致性定义为相应建议的长度（是输入长度的函数）。

定义 1.12（接受建议） 称算法 A 利用长度为 $l: \mathbb{N} \rightarrow \mathbb{N}$ 的建议计算函数 f ，如果存在一个无限长序列 $(a_n)_{n \in \mathbb{N}}$ ，满足：

(1) 对任意 $x \in \{0,1\}^*$ ，有 $A(a_{|x|}, x) = f(x)$ ；

(2) 对任意 $n \in \mathbf{N}$, 有 $|a_n| = l(n)$,

序列 $(a_n)_{n \in \mathbf{N}}$ 称为建议序列。

注意: 任何一个电路复杂度为 s 的函数都可利用长度为 $O(s \log s)$ 的建议求解, 其中包含对数因子是由于含 v 个顶点和 e 条边的图可以用一个长度为 $2e \log_2 v$ 的串来描述。注意使用建议的机器模型允许比 1.2.4.1 节中更尖锐的界限: 每个函数可用长为 l 的建议求解, 其中 $l(n) = 2^n$, 且某些不可计算的函数可用长为 1 的建议求解。

定理 1.13 (建议的功能) 存在一些函数, 可用一比特的建议求解, 但是无建议时却无法求解。

证明: 从任意一个不可计算的布尔函数 $f: \mathbf{N} \rightarrow \{0,1\}$ 开始, 考虑函数 f' , $f'(x) = f(|x|)$ 。

注意: f 可以图灵归约为 f' (如, 对于输入 n , 向 f' 发出长为 n 比特的查询并返回结果)。^①因此, 没有建议时 f' 不可解。另一方面, 利用建议序列 $(a_n)_{n \in \mathbf{N}}$, 其中 $a_n = f(n)$, 则 f' 是易解的, 此时求解 f' 的算法只需输出建议即可 (事实上, 对任意 $x \in \{0,1\}^*$, $a_{|x|} = f(|x|) = f'(x)$)。 ■

1.2.4.3 受限的计算模型

布尔电路模型 (见 1.2.4.1 节) 可以引入计算设备的许多自然子类。以下简要介绍了几个这样的子类。进一步的细节可见附录 B.2。由于本书其余章节中大量涉及到各种类型的布尔公式, 所以我们建议不要忽略下面的内容。

布尔公式

布尔电路中的非退化顶点 (一般来说) 允许出度为任意值。这意味着同一个中间值可以重复使用而无需再次计算 (且这样做只对电路的规模复杂度增加一个单位)。这种对中间结果的“自由”使用在布尔公式中是不允许的, 布尔公式是指所有非退化顶点出度均为 1 的布尔电路。这意味着布尔公式对应的图是一棵树 (见附录 G.2), 且对该图的遍历 (并按照遍历顺序记下经过的顶点标号) 可以写成关于布尔变量的表达式。事实上, 这里允许为不同的输入端分配同一标号, 从而允许同一变量多次出现。在一般电路中, 人们对这些受限电路的规模感兴趣 (即计算各种函数的类的规模)。需要指出, 虽然已知某些简单函数 (如奇偶校验) 具有二次的规模下限, 然而这些函数的电路复杂度却是线性的。图 1.4 描绘了这种矛盾。

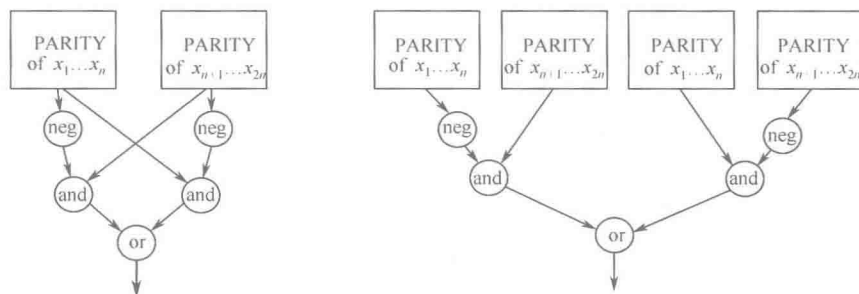


图 1.4 奇偶校验电路及公式的递归构造

CNF 与 DNF 范式

一种受限的布尔公式类型是合取范式 (Conjunctive Normal Form, CNF)。这种公式由一些子句的合取构成, 其中每个子句都是文字的析取, 而这些文字或者为变量, 或者为变量取非。

^① 事实上, 该图灵归约并非充分 (即运行于 $|n| = \log_2 n$ 的指数时间内), 但在这里并不重要。

从而,这种公式可以表示为分层的无界扇入电路,其中第一层由非门构成,对输入变量取非,第二层由或门构成,计算部分输入变量及输入变量取非的逻辑或。第三层包含单个的与门,计算第二层中求出的值的逻辑与。注意:每个布尔函数都可用具有指数规模的 CNF 范式类来计算,且 CNF 范式的规模可以是计算同一函数的一般电路(如奇偶校验电路)规模的指数倍。对于常数 k ,称一个公式属于 k -CNF,如果其 CNF 范式中析取式的规模最多为 k 。布尔公式的另一种受限类型是析取范式(Disjunctive Normal Form, DNF)。这种公式由文字合取式的析取构成,如果每个合取式中最多包含 k 个变量,则称该范式属于 k -DNF。

常数深度的电路

电路具有一种“自然结构”(即图)。这种结构中一个自然的参数就是电路的深度,定义为由任意源出发到任意终端结点的最长路径。人们尤其对具有无界扇入的常数深度电路感兴趣。我们指出已知的计算简单函数(如奇偶校验)的这种电路具有亚指数的规模下限。

单调电路

电路模型中还考虑了单调的计算设备:单调电路是指那些只包含单调门(如计算 $\wedge$ 和 $\vee$ 的门,但没有非门(即 $\neg$ 门))的电路。当然,单调电路只能计算单调函数,这里称一个函数 $f: \{0,1\}^n \rightarrow \{0,1\}$ 是单调的,如果对任意 $x \preceq y$,有 $f(x) \leq f(y)$ (其中 $x_1 x_2 \cdots x_n \preceq y_1 y_2 \cdots y_n$ 当且仅当在任意位置 i ,有 $x_i \leq y_i$)。这里自然会产生一个问题:就单调函数而言,仅仅使用单调电路是否会带来实质性的损失。回答是肯定的:存在一些电路复杂度为多项式的单调函数,但是需要亚指数规模的单调电路来计算。

1.2.5 复杂性类

复杂性类是对计算问题的分类。典型地,在固定了三个参数后,可以定义如下的类:

(1) 一类计算问题(见 1.2.2 节)。事实上,大多数类都是关于判定问题的,但是也会考虑搜索问题、承诺问题及其他类型的问题。

(2) 一种计算模型,可以是一致的(见 1.2.3 节)或非一致的(见 1.2.4 节)。

(3) 一种复杂性量度及一个界限函数(或函数集合),将前两项中的计算类型放在一起,即针对那些复杂度不超过规定函数的计算类(或函数集合)。例如,在 1.2.3.5 节,我们提到了时间复杂度及空间复杂度,可用于任意一致的计算模型。我们还提到了多项式时间的计算,这些计算的时间复杂度(作为一个函数)不超过某个多项式(即多项式函数集中的某个成员)。

最常见的复杂性类是关于判定问题的,且有时被定义为对集合的分类而非对相应判定问题的分类。换言之,人们常常称一个集合 $S \subseteq \{0,1\}^*$ 属于类 C ,而不说 S 的成员判定问题属于 C 。类似地,在搜索问题中,常对关系进行分类而不是对相应搜索问题分类(即称 $R \subseteq \{0,1\}^* \times \{0,1\}^*$ 属于类 C ,意为 R 的搜索问题属于 C)。

本章注释

关于一致和非一致的计算设备,最卓越的工作是两篇文章,即图灵的论文^[225],其中介绍了图灵机模型,以及仙农的论文^[203],其中介绍了布尔电路。

除了引入图灵机模型,并指出其对应于直观上计算的概念之外,图灵的文章^[225]还引入了通用机器,并证明了一些问题的不可判定性(如停机问题的不可判定性)。

丘奇—图灵（Church-Turing）定理来自于丘奇^[55]和图灵^[225]的工作。这两篇文章中都引用该定理来证明一个事实，即某些问题不能用图灵机求解蕴含了该问题也不能用任意“合理”的计算模型求解。RAM 模型来自于冯·诺伊曼（von Neumann）的报告^[234]。

将高效计算与多项式时间算法相关联最早见于 Cobham^[57]和 Edmonds 的工作^[70]。有趣的是，Cobham 的出发点是要对高效算法给出一种哲学上的合理解释，而 Edmonds 的出发点是要清晰地解释为什么某些算法在实际中是“好的”。

参考文献[192]中证明了莱斯定理，波斯特对应问题的不可判定性在参考文献[181]中得到证明。接受建议的机器的形式化定义（以及关于电路模型的等价性）最早见于参考文献[139]。

第2章

P、NP 和 NP-完全性

有好些人提笔作书，述说在我们中间所成就的事，是照传道的人从起初亲眼看见，又传给我们的，这些事我既从起头都详细考察了，就决意要按着次序写给你，使你知道所学之道都是确实的。

新约——路加福音第1章：1-4

本章主要讲述 P-vs-NP 问题以及 NP-完全性理论。另外还包括多项式时间归约的一般概念（特别强调了自归约性），NP 中既非 NP-完全也不属于 P 类的问题的存在性，coNP 类，优化的搜索问题，以及承诺问题。

内容提要

大体上讲，P-vs-NP 问题主要考虑其解的正确性可以被高效验证的搜索问题（即存在有效算法，给定问题实例的解，能判定该解是否正确）。这种搜索问题相当于 NP 类，而 P-vs-NP 问题是指这样的搜索问题能否被高效求解（即存在高效算法，能找到给定实例的正确解）。因此，P-vs-NP 问题也可表述为“是否求解比验证解的正确性更困难？”

该问题还有一个用判定问题阐述的定义，针对着可以高效验证（验证过程相对较短）的断言。这样的断言集合对应着 NP 类，此时的 P-vs-NP 问题是指是否可以快速地找到这些断言的证明（即是否存在高效算法，在给定一个断言时，能判定其正确性，并且/或者找到对其正确性的证明）。因此，P-vs-NP 问题可以描述为“是否寻找证明比验证证明的正确性更困难”，这就是说，证明是否比验证更难（或者证明是否真的有价值）。

事实上，普遍认为对这两种等价表述的回答是，寻找（或发现）比检查（或验证）更难。也就是说，P 不同于 NP。这种自然的猜想至今未得到证明，这一事实似乎是复杂性理论研究中的最大失败之一。然而，笔者的观点是，这种失败感并不太恰当。因为目前在任何情况下，面对 NP 类中的困难问题，我们并不期望能（无条件地）证明这个问题不属于 P，而是希望基于 P 不同于 NP 的假设来（有条件地）证明该问题不属于 P。其逆否命题是证明该问题属于 P，从而 NP 中任意一个问题都属于 P（即 $NP=P$ ）。这才是 NP-完全性理论的诱人之处。

NP-完全性理论的基础是归约，它描述了计算问题之间的关系。大体上讲，称一个问题归约到另一个问题，如果在给定解后者的（高效）算法时，前者也可以高效地解决，从而，第一个问题并不比第二个问题更难。称一个（NP）问题是 NP 完全的，如果 NP 中任意问题都可以归约到该问题的话。因此，整个 NP 类的命运（是否包含于 P 中）取决于每个单独的 NP-完全问题。特别地，证明一个问题是 NP-完全的意味着该问题不属于 P，除非已经证明了

$NP=P$ 。令人惊奇的是, NP -完全问题确实存在, 而且有上百个来自数学和自然科学不同领域的计算问题都是 NP -完全问题。

我们强调 NP -完全问题并不是 NP 中仅有的困难问题 (即如果 $P \neq NP$, 则 NP 中还包含既非 NP -完全也非 P 的问题)。我们还将指出 P -vs- NP 问题的内在含义并不是开发复杂的算法或证明这种算法不存在, 而是归结为对某个已经算法的分析, 即对于 NP 中的任何问题, 我们会提出一个最优的搜索算法, 但是对其时间复杂性一无所知。

教学注释: 事实上, 我们建议同时使用搜索问题术语和判定问题术语来阐述 P -vs- NP 问题。另外, 在后一种情况下, 最好明确地利用证明系统的术语来引入 NP 。对于 NP -完全性理论, 建议要强调 NP -完全问题的存在性。

预备知识

假设读者熟悉搜索与判定问题 (见 1.2.2 节)、算法 (见 1.2.3 节) 及其时间复杂度 (见 1.2.3.5 节) 的概念, 另外本章还涉及到预言机 (见 1.2.3.6 节) 的相关概念。

内容组织

2.1 节提出 P -vs- NP 问题的两种表述。2.2 节阐述归约的一般概念, 并强调了归约在 NP -完全性理论之外的其他领域的用途。2.3 节介绍 NP -完全性理论, 2.4 节介绍三个相对前沿的话题 (即承诺问题的结构、 NP 中最优搜索算法的存在性以及 $coNP$ 类)。

教学注释: 与其他章节相比, 本章的教学注释是最多的。这反映出作者更关心这些基础知识的教学。作者认为, 对于本章所覆盖的基础知识, 人们通常采用的教学方法也许并不妥当, 因为在教学中没有揭示事物的基本特性。

2.1 P -vs- NP 问题

人们的日常经验是解一个问题要比验证解的正确性更困难。这种经验仅仅是巧合还是体现了生活中一个基本事实 (或世界的一个属性)? 这就是 P -vs- NP 问题的本质, 其中 P 表示可以高效求解的搜索问题, 而 NP 表示其解可以被高效验证的搜索问题。

P -vs- NP 问题代表的另一个自然问题是, 是否证明定理比验证证明的正确性更困难。换言之, 集合的成员判定是否比通过证明来确认集合成员更困难。此时, P 表示可以被高效求解的判定问题, 而 NP 表示可以其成员证明可以被高效验证的集合。

2.1.1 节和 2.1.2 节分别严格描述了 P -vs- NP 问题的上述两种含义。2.1.3 节证明了两种表述的等价性, 2.1.6 节进一步讨论了人们通常相信的“ P 不同于 NP ”。以下我们从对高效计算概念的回顾开始。

教学注释: 大多数学生曾经听说过 P 与 NP , 但是对于 P -vs- NP 问题的根本含义, 许多人并未很好地理解。这种不幸局面是由于对 NP 问题使用标准的技术定义而造成的 (其中使用了一种虚构的、令人困惑的设备, 称为非确定的多项式时间机器)。相反, 我们在以下章节中提倡使用更复杂的定义 (2.1.1 节和 2.1.2 节中将详细介绍), 它可以更清晰地把握 NP 的本质特征。

高效计算

回顾上一章中将高效计算与多项式时间算法相关联。^①这种关联是合理的，因为多项式是一种缓慢增长的函数类，并对算法的自然合成运算封闭。另外，多项式时间算法类独立于具体的计算模型，只要模型是“合理的”即可（见 Cobham-Edmonds 定理）。这两个问题在 1.2.3.5 节中均已讨论过。

对问题实例表达方式的进一步注释：如 1.2.2.3 节所述，自然界中许多（搜索和判定）问题都可以用承诺问题术语来更自然地表达（见 2.4.1 节），而在承诺问题中，实例可能是 $\{0,1\}^*$ 的子集，而非 $\{0,1\}^*$ 本身。例如，图论中的计算问题假设将图简单地编码为字符串，但这种编码过程并非满射（即并非所有的串都是某个图的编码），从而并非所有的字符串都是合理的实例。然而，在这些情况中，合理实例集（即图的编码）可以被高效识别（即可以在多项式时间内判定集合成员）。因此，人为地将实例集扩展到所有可能串（并假定所有“哑”实例对应着平凡解）并不改变原始问题的复杂性。2.4.1 节将进一步讨论这个问题。

2.1.1 搜索版本：求解与检验

教学注释

（1）复杂性理论是如此地习惯于讨论判定问题，以至于似乎忘记了搜索问题也与判定问题一样属于自然问题。另外，对许多外行而言，搜索问题可能比判定问题更加常见：典型地，人们更习惯于寻找问题的解而不是仅仅考虑问题是否有解。因此，我们建议在开始讲述 P-vs-NP 问题时使用搜索问题术语。必须承认，此时表达起来会更复杂，但这样做是值得的。

（2）为了体现搜索版本的重要性，同时又使用不那么复杂的表达，我们选择引入对应于 P 和 NP 的两个搜索类的符号：这两个类被记作 PF 和 PC（分别表示多项式时间可寻找和多项式时间可验证）。教师可以使用相应于 P 和 NP 的更具提示性的概念和术语（如 P-search 和 NP-search），实际上我们在一些关于动机的讨论中已经这样做了（特别是在本书的后续章节中）。（虽然如此，我们认为从长远来看，使用标准记号更有利于学生学习）。

计算机科学在很大程度上更倾向于求解各类搜索问题（如定义 1.1）。这些问题包括解线性（或高次）方程组、求给定整数的素因子、求图的生成树、求旅行售货员的最短路径，以及求满足各种约束条件的机器工作日程表等。此外，解搜索问题对应于人们日常所说的“求解问题”，因此自然会得到普遍关注。本节考虑这样一个问题：哪些搜索问题可以被高效地求解。

存在这样一类搜索问题，它们不能被高效求解的原因是，相对于问题实例而言，解的长度太长了。此时，仅仅将解输入到计算机中也是一个效率极低的行为。而我们关注的搜索问题不在此列。也就是说，我们只考虑那些解的长度是问题实例长度的多项式的搜索问题。由于搜索问题对应于二元关系（见定义 1.1），因此我们讨论的重点是多项式界的关系。

定义 2.1（多项式界关系） 二元关系 $R \subseteq \{0,1\}^* \times \{0,1\}^*$ 是多项式界的，如果存在一个多项式 p ，使得对任意 $(x,y) \in R$ ，有 $|y| \leq p(|x|)$ 。

这里 $(x,y) \in R$ 的含义为： y 是问题实例 x 的解，而 R 表示问题本身。例如，在求整数的素因子时，关系 R 是指，如果 y 是 x 的素因子，则 $(x,y) \in R$ 。

^① 本章我们将（多项式时间的）确定算法作为高效计算的基本模型。从更自由的角度看，高效计算还应该包含第 6 章中的（多项式时间）概率算法。我们强调本章讨论的最重要的事实和问题对于概率多项式时间算法也成立。

对于多项式界关系 R 而言, 能否高效地找到给定实例 x 的解 y (即找到 y , 使得 $(x, y) \in R$) 是有意义的。解 (即 y) 长度的多项式界保证了不会仅仅由于解的长度而导致对这一问题的否定回答。

2.1.1.1 作为自然搜索问题的 P 类

我们感兴趣的是可以被高效求解的搜索问题类 (如果存在解)。将讨论范围局限于多项式界关系, 则可以确定出相应搜索问题 (或二元关系) 的基本类型, 记作 $\mathcal{PF}$ (表示 “在多项式时间找到”)。($\mathcal{PF}$ 与 P 类的标准定义之间的关系将在 2.1.3 节和 2.2.3 节讨论。) 以下定义是定义 1.1 中解搜索问题的形式化描述。

定义 2.2 (可高效求解的搜索问题)

- 称多项式界关系 $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ 对应的搜索问题是可以高效求解的, 如果存在多项式时间算法 A , 使得对任意 x , 有 $A(x) \in R(x) \stackrel{\text{def}}{=} \{y : (x, y) \in R\}$ 当且仅当 $R(x)$ 非空。进而, 如果 $R(x) = \emptyset$, 则 $A(x) = \perp$, 表示 x 无解。

- 用 $\mathcal{PF}$ 表示可以高效求解的搜索问题类 (对应于多项式界关系)。也就是说, 如果 R 为多项式界关系, 并且存在多项式时间算法, 给定 x 时能求出 y 使得 $(x, y) \in R$ (或断定这样的 y 不存在), 则 $R \in \mathcal{PF}$ 。

注意到 $R(x)$ 表示问题实例 x 的合法解集, 因此, 只要该实例有解 ($R(x)$ 非空), 就要求算法 A 求出一个合法的解 (即满足 $A(x) \in R(x)$)。另一方面, 如果实例 x 无解 (即 $R(x) = \emptyset$), 则显然 $A(x) \notin R(x)$ 。这种情况有一个附加的条件即 $A(x) = \perp$ (定义 1.1 中也有)。从而, 算法 A 总是能输出正确答案, 如果问题实例有解, 则输出一个合法解, 否则指出该实例无解。

我们已经定义了一个基本的问题类, 并且知道自然界中许多问题都属于这个类 (如解有理系数线性方程、求图的最佳匹配等)。然而, 必须承认, 我们仍未很好地理解这类问题的本质 (无法给出这一类中许多自然问题的一般特征)。这种情况在复杂性理论中很常见, 其成因似乎要归咎于在定义复杂性类时使用 “外部行为” (潜在的算法) 术语而非 “内部结构” (问题本身) 术语。再回到 $\mathcal{PF}$ 类, 注意到虽然其中包含了自然界中许多搜索问题, 但还有许多搜索问题不属于 $\mathcal{PF}$ 。以下给出一个包含大量这种问题的类。

2.1.1.2 作为另一类自然搜索问题的 NP 类

自然界中的搜索问题有这样一个特征, 其合法解是可以被高效识别的。即给定问题 R 的一个实例 x , 以及候选解 y , 可以高效地确定 y 是否为 x 的合法解 (相应于问题 R , 即是否有 $y \in R(x)$)。所有的这种搜索问题在本质上属于同一类, 因为除非可以识别给定解是否合法, 否则又何必考虑这个解呢? 进一步地, 这个类是由 $\mathcal{PF}$ 中候选问题形成的一个自然类, 因为为了讨论求出正确解的复杂性, 一个自然的 (尽管并非绝对的) 先决条件似乎是高效识别正确解的能力。

现在重新把注意力局限于多项式界关系, 并考虑由成员可以被高效判定的关系所构成的类。我们强调考虑的是判定一个给定的形如 (x, y) 的对是否属于某个关系 R , 而不是判定 x 是否属于集合 $S_R \stackrel{\text{def}}{=} \{x : R(x) \neq \emptyset\}$ 。(以下对于 NP 的定义与 NP 的标准定义之间的关系将在 2.1.3 节~2.1.5 节以及 2.2.3 节中讨论。)

定义 2.3 (具有可高效验证解的搜索问题)

- 称多项式界关系 $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ 的搜索问题具有可高效验证的解, 如果存在一个多

项式时间算法 A , 使得对任意的 x 和 y , $A(x, y) = 1$ 当且仅当 $(x, y) \in R$ 。

• 用 PC (Polynomial-time Check, 表示“多项式时间可验证”) 表示对应于具有可高效验证解的多项式界二元关系的搜索问题类。就是说, 如果以下两个条件满足, 则 $R \in PC$:

- (1) 对某个多项式 p , 如果 $(x, y) \in R$, 则 $|y| \leq p(|x|)$;
- (2) 存在一个多项式时间算法, 给定 (x, y) , 可以判定是否 $(x, y) \in R$ 。

PC 类中包含了数以千计的自然问题 (如在旅行售货员问题中, 求长度不超过某个给定阈值的旅行路线, 或者求给定合数的素因子分解等)。每个这类问题的解都可以被高效地验证 (如给定一条旅行路线, 很容易求出其长度并检查是否超过了给定的阈值)。^①

为了研究何种问题属于 PF 类, 自然可将讨论范围限于 PC 类, 因为高效识别正确解的能力是讨论求解复杂度的先决条件。虽然如此, 我们要提醒读者注意, PF 中也包含 (非自然的) 不属于 PC 的问题 (见习题 2.1)。

2.1.1.3 搜索形式的 P-vs-NP 问题

是否 PC 中所有的搜索问题都属于 PF ? 如果可以高效验证对应于某个 (多项式界) 关系 R 的解的正确性, 则是否 R 的搜索问题一定都能高效求解? 换言之, 如果易验证一个给定实例的给定解的正确性, 则对给定实例求解是否也是容易的?

如果 $PC \subseteq PF$, 则意味着只要给定实例的解 (的正确性) 可以被高效验证, 则也能高效地求解 (在只给定该实例的条件下)。就是说所有合理的搜索问题 (即 PC 中所有问题) 都是易解的。当然, 这与人们的直觉 (以及日常经验) 相悖, 因为某些合理的搜索问题确实很难解。另外, 在这种情况下, “解一个问题” 的概念会失去其本来含义 (因为此时求一个解不比验证其正确性更困难)。

另一方面, 如果 $PC \setminus PF \neq \emptyset$, 则存在一些合理的搜索问题 (即 PC 中某些问题) 是难解的。这符合我们的直觉认识: 某些合理问题是易解的, 而另一些则是难解的。另外, 它也重新证实了直觉上 “求解问题” 与 “验证答案” 这两个概念间的差异 (断定在某些情况下 “求解” 比 “验证” 要难得多)。

2.1.2 判定版本: 证明与验证

以下我们将看到, 对搜索问题的研究 (如 PC -vs- PF 问题) 可以 “简化” 为对判定问题的研究。后者表述起来不像前者那样复杂, 因而成为复杂性理论研究的重点 (并通过上述简化过程保持着与搜索问题的相关性)。因此, 对判定问题的研究提供了研究搜索问题的简便方法。例如, 对判定布尔表达式可满足性的复杂度的研究, 提供了一种简便方法来研究寻找这些表达式的可满足性赋值的复杂度。

然而, 要强调的是, 判定问题本身也是很有趣的 (除了能在搜索问题研究中发挥作用以外)。毕竟许多人对真理感兴趣, 所以判定一个给定的断言是否为真也属于自然的计算问题。特别是, 判定一个给定对象 (如布尔表达式) 是否具有某种预先定义的性质 (如可满足性) 构成了一类有趣的计算问题。 P -vs- NP 问题可归结为针对与 NP 类相关联的广泛且自然的属性类, 求解上述问题的复杂性。后者是指那些具备 “高效证明体系” 的属性, 可以验证一个给定对象具有预先定义性质的断言 (即属于某个预定义的集合) 是否成立。后面将提到, 我们所说的 P -vs- NP 问题主要是指能高效证明的性质也能被高效地判定 (无需证明)。下面澄清这

^① 在旅行售货员问题 (TSP) 中, 实例为表示城市间距离的矩阵及一个阈值, 目标是求一条旅行路线, 经过所有城市, 且总长度不超过阈值。

些概念。

对象的性质被形式化地定义为所有可能对象的子集（即一种性质与具有该性质的对象构成的子集相关联）。例如，整数的素性与素数集相关，而图的连通性（或具有哈密尔顿回路，Hamiltonian path）与所有连通图（或哈密尔顿图）相关。因此，我们重点关注集合成员的判定（如定义 1.2）。P-vs-NP 问题的标准陈述是指两种判定问题类的等价性，这两类问题记作 P 和 NP（分别在 2.1.2.1 节和 2.1.2.2 节中给出了定义。）

2.1.2.1 作为自然判定问题的 P 类

人们显然对能高效求解的判定问题感兴趣。这类问题习惯上记作 $\mathcal{P}$ 类（表示“多项式时间”，Polynomial-time）。以下给出解判定问题（见定义 1.2）的形式化定义。

定义 2.4（可高效求解的判定问题）

- 判定问题 $S \subseteq \{0,1\}^*$ 可以被高效求解，如果存在一个多项式时间算法 A 使得对任意的 x ， $A(x)=1$ 当且仅当 $x \in S$ 。
- 用 $\mathcal{P}$ 表示可以高效求解的判定问题。

与定义 2.2 相同，这里定义了一个基本的问题类，其中包含许多自然问题（如判断一个给定的图是否连通），但是没有全面涵盖其本质内容（即对于这个类中的许多自然问题，该定义尚不能刻画其属于该类的特征）。事实上，存在许多判定问题无法辨别其是否属于 $\mathcal{P}$ 类，下面我们将给出一大类这种问题。这类判定问题被记作 $\mathcal{NP}$ （在 2.1.5 节将说明为什么叫这个名称）。

2.1.2.2 NP 类及 NP-证明系统

将 $\mathcal{NP}$ 看作是具有可高效验证证明系统的判定问题。不严格地讲，如果一个集合 S 中的实例具有可高效验证的成员证明（即系统认为该证明合理并接受），而不属于 S 的实例没有高效证明，则称集合 S 具有证明系统。事实上，这里的证明被定义为可由（高效）验证程序接受的串（与实例相伴）。称 V 是 S 中的成员验证程序，如果它满足以下两个条件：

（1）完备性。正确的断言具有有效证明，即被 V 接受的证明。要记住的是，这里的断言是指关于 S 中成员的断言，即对任意 $x \in S$ ，存在一个串 y ，使得 $V(x,y)=1$ （即 V 接受 y 作为 x 属于 S 的有效证明）。

（2）合理性。错误的断言没有有效证明，即对任意 $x \notin S$ 及所有的串 y ，有 $V(x,y)=0$ ，表明 V 会拒绝 y 作为 x 属于 S 的成员证明。

注意到合理性条件体现了验证程序的“安全性”，即验证程序不会被（任何事物）欺骗而做出错误断言。完备性条件则体现了验证程序的“生存能力”，即验证程序在有充分证明时，能确认任意一个正确断言的有效性（我们强调，通常证明系统是用其验证程序来定义的，后者必须满足完备性和合理性条件）。在这里讨论的重点是使用相对较短的证明（即证明的长度为相应断言长度的多项式）^①的高效验证程序。

在给出正式定义之前，先讨论几个例子。例如，全体哈密尔顿图的集合具有这样的验证程序：给定一对 (G, π) ，当且仅当 π 是图 G 中的哈密尔顿回路时，程序返回“接受”。此时 π

^① 作为前面注释的继续，本章我们考虑的是（多项式时间）确定性验证程序，因此，这里所说的完备性和合理性条件都是无错误的。相反，在第 9 章，会讨论各种类型的概率（多项式时间）验证程序，以及概率完备性和合理性条件。通常在这两个条件中有效验证被解释为由运行时间为断言的多项式之内的程序所验证。本章使用等价的形式化定义，将运行时间看作是断言+证明总长度的函数，但证明长度必须为断言长度的多项式。

就成为 G 是哈密尔顿图的一个证明。注意到这种证据相对较短（即路径长度比图的描述要短）且易于验证。显然，这个证明系统满足上述完备性和合理性条件。在布尔表达式的可满足性问题中，给定一个表达式 Φ 和一种真值赋值 τ ，验证程序计算 Φ （关于 τ ），当且仅当经过化简后的布尔表达式返回值为“真”时，验证程序返回“接受”。此时， τ 就成为 Φ 可满足的一个证明，该证明长度较短且易于验证。

定义 2.5（可高效验证的证明系统）

• 称判定问题 $S \subseteq \{0,1\}^*$ 具有可高效验证的证明系统，如果存在一个多项式 p 以及多项式时间算法 V ，使得以下两个条件成立：

(1) 完备性。对任意 $x \in S$ ，存在长度至多为 $p(|x|)$ 的 y ，使得 $V(x, y) = 1$ （这样的串 y 被称为 $x \in S$ 的一个 NP-证据（NP-witness））。

2. 合理性。对任意 $x \notin S$ 及任意的 y ，有 $V(x, y) = 0$ 。

从而 $x \in S$ 当且仅当存在长度至多为 $p(|x|)$ 的 y ，使得 $V(x, y) = 1$ 。

此时，我们称 S 具有 NP-证明系统，并称 V 为其验证程序（或证明系统本身）。

• 用 $\mathcal{NP}$ 表示具有可高效验证的证明系统的判定问题类。

注意：“NP-证据”这个术语是通用的。^①有时把 V （或者由 V 接受的对集合）称为 S 的证据关系（Witness Relation）。我们强调同一集合 S 可能有许多不同的 NP-证明系统（见习题 2.2），而有些情况下这种差别并不是人为造成的（见习题 2.3）。

教学注释：利用定义 2.5，可以很容易地证明一个判定问题是否属于 NP。只需设计出关于其成员的足够多的 NP-证据即可，而这种设计是很直接便捷的，因为判定问题通常被阐述为是否存在易被高效验证的结构（或对象）。例如，SAT 被定义为可满足的布尔表达式集合，意思是判定是否存在可满足的真值赋值。事实上，我们可以高效地检查给定的赋值是否满足给定的表达式，就是说，SAT 的 NP-证明体系（验证程序）是存在的。

注意到对 $\mathcal{PC}$ 中任意一个搜索问题 R ，有解的实例集（即集合 $S_R \stackrel{\text{def}}{=} \{x: R(x) \neq \emptyset\}$ ）属于 $\mathcal{NP}$ 。特别地，对任意 $R \in \mathcal{PC}$ ，考虑验证程序 V ，使得 $V(x, y) \stackrel{\text{def}}{=} 1$ 当且仅当 $(x, y) \in R$ ，注意到这个条件可以在 $\text{poly}(|x|)$ 时间内判定。因此， $\mathcal{PC}$ 中所有的搜索问题都可看作是寻找（可高效验证的）证明的问题（即寻找属于有解的实例集成员的 NP 证据）。另一方面，任意一个 NP-证明系统均能诱导出 $\mathcal{PC}$ 中的一个搜索问题：即对于给定实例寻找正确证明（即 NP-证据）的问题（就是说，验证程序 V 诱导出对应于 $R = \{(x, y): V(x, y) = 1\}$ 一个搜索问题）。因此， $S \in \mathcal{NP}$ 当且仅当存在 $R \in \mathcal{PC}$ 使得 $S = \{x: R(x) \neq \emptyset\}$ 。

教学注释：最后一段提出了另一种简单方法来证明判定问题属于 $\mathcal{NP}$ ：只需将其对应到相应的搜索问题即可。关键在于（ $\mathcal{NP}$ 中的）判定问题被解释为相应的搜索问题是否有解。例如，SAT 问题的含义是对于给定的布尔表达式，是否存在可满足的赋值，而相应的搜索问题是找到这样一种赋值。但是在所有情况下，解的正确性都是易于检查的；就是说，相应的搜索问题属于 $\mathcal{PC}$ ，而这暗示着判定问题属于 $\mathcal{NP}$ 。

^① 在多数情况下，并不需要明确定义 V ， V 的含义由上下文来理解。许多文献并不把 V 称为一个证明系统（或这类系统的验证程序），虽然这个术语是最充分的。

观察到 $P \subseteq NP$ 是成立的：关于集合 $S \in P$ 成员的验证程序只需忽略所谓的 NP-证据，并运行由假设 $S \in P$ 所保证的判定程序；即 $V(x, y) = A(x)$ ，其中 A 就是前面所说的判定程序。其实后一个验证程序是滥用了术语（因为其中证明并不起作用）；然而，它仍是一个合法的程序。接下来我们将看到 P-vs-NP 问题的意思是，能否将这种证明可忽略的验证程序用于任意一个具有可高效验证的证明系统的集合（实际上，如果假设 $P \subseteq NP$ ，则 P-vs-NP 问题可归结为 $NP \subseteq P$ 是否成立）。

2.1.2.3 判定形式的 P-vs-NP 问题

NP-证明是否真的没有用处？换言之，对任意可高效验证的证明系统，在不参考证明时，断言的正确性是否易于判定？如果是这样，则证明将毫无意义，因为与直接确定断言的正确性相比，它们根本不具备任何优势。 $P \neq NP$ 的猜想认为证明是有用的： NP 中存在一些集合，不存在判定其成员的多项式时间算法，从而对这些集合，得到（某些实例的）成员证明是有用的（因为我们不能高效地确定其成员）。

上文将 $P \neq NP$ 看作是获得证明比单纯靠判定更具优势。即 $P \neq NP$ 认为（在某些情况下）验证比判定更容易。一个略有不同的观点则认为 $P \neq NP$ 表示寻找证明比验证其有效性更困难。这是因为，对任意具有 NP-证明系统的集合 S ，高效地找到关于该系统的成员证明（即对任意给定的 $x \in S$ ，求 S 成员的 NP-证据），蕴含着能对 S 进行成员判定。因此，对 $S \in NP \setminus P$ ，求 S 的成员证明比验证这种证明的正确性（可在多项式时间内完成）更困难。

2.1.3 两种表示的等价性

前面已多次暗示过，P-vs-NP 问题的两种表现形式是等价的。即任意具有可高效检查的解的搜索问题也可在多项式时间内求解（即 $PC \subseteq PF$ ），当且仅当任意一个具有 NP-证明系统的集合的成员判定问题可在多项式时间内完成（即 $NP \subseteq P$ ）。结合 $P \subseteq NP$ （而 PC 不包含于 PC 中（习题 2.1）），可以证明如下结论。

定理 2.6 $PC \subseteq PF$ 当且仅当 $P = NP$ 。

证明：一方面，假设包含关系对于搜索问题成立（即 $PC \subseteq PF$ ）。我们将证明这蕴含了对 NP 中任意集合，都存在找到其 NP-证据的高效算法，这又表明该集合属于 P 。具体而言，设 S 是 NP 中任一集合， V 为相应的验证算法（即满足定义 2.5 的条件），则 $R \stackrel{\text{def}}{=} \{(x, y) : V(x, y) = 1\}$ 是 PC 中一个多项式界限的关系，且由假设，其对应的搜索问题可在多项式时间内解决（即 $R \in PC \subseteq PF$ ）。将解 R 的搜索问题的多项式时间算法记作 A ，则确定 S 的成员是容易的。即对输入的 x ，当且仅当 $A(x) \neq \perp$ 时输出 1，后者成立当且仅当 $A(x) \in R(x)$ ，即当且仅当 $R(x) \neq \emptyset$ （等价地，有 $x \in S$ ）。因此，就证明了 $NP \subseteq P$ （从而 $NP = P$ ）。

另一方面，假设 $NP = P$ ，我们将证明这蕴含了存在高效算法，可以判定一个给定串 y' 是否为 PC 中某个搜索问题的给定实例 x 的前缀，从而可以得到求解的高效算法。具体地，设 R 为 PC 中任意一个搜索问题，则集合 $S'_R \stackrel{\text{def}}{=} \{(x, y') : \exists y'', (x, y', y'') \in R\}$ 属于 NP （因为 $R \in PC$ ），从而 S'_R 属于 P （由 $NP = P$ 的假设）。这样，通过逐比特地扩展某个可能解的前缀（同时使用确定程序来判定该前缀是否正确），就得到了解 R 的搜索问题的多项式时间算法。即对于输入的 x ，首先检查是否有 $(x, \lambda) \in S'_R$ 并当 $(x, \lambda) \notin S'_R$ 时输出 $\perp$ （表明 $R(x) = \emptyset$ ）。下面继续迭代，并保持 $(x, y') \in S'_R$ 不变。在每次迭代中，当 $(x, y'_0) \in S'_R$ 时令 $y' \leftarrow y'_0$ ，而当 $(x, y'_1) \in S'_R$ 时令

$y' \leftarrow y'_i$ 。如果二者均不成立（在至多多项式次迭代之后就会发生），则当前的 y' 满足 $(x, y') \in R$ 。因此，对任意 $R \in PC$ ，必有 $R \in PF$ ，从而 $PC \subseteq PF$ 。 ■

思考：定理 2.6 证明中的第一部分将 NP 中每个集合 S 与关系 R （在 PC 中）相关联。具体地， R 中包含所有这样的对 (x, y) ，其中 y 是 $x \in S$ 的一个 NP-证据。因此， R 的搜索问题即为当给定 x 作为输入时，求这样的 NP-证据。事实上， R 称为 S 的证据关系，而解 R 的搜索问题时必须考虑 S 的成员判定。因此， $R \in PC \subseteq PF$ 蕴含了 $S \in P$ 。在证明的第二部分，将每个 $R \in PC$ 与集合 S'_R （在 NP 中）相关联，但是 S'_R 比集合 $S_R \stackrel{\text{def}}{=} \{x: \exists y, \text{s.t. } (x, y) \in R\}$ 更具“表现力”（它使 R 成为其（ S_R 的）证据关系）。具体地， S'_R 中包含对 (x, y') 编码后的串，其中 y' 是 $R(x) = \{y: (x, y) \in R\}$ 中某个串的前缀。一个重要的事实是，对 S'_R 进行成员判定时必须考虑解 R 的搜索问题，即 $S'_R \in P$ 蕴含着 $R \in PF$ 。

结论：定理 2.6 证明了重点讨论 P-vs-NP 的判定版本这一传统做法的合理性。事实上，该问题的两种定义是等价的，只需研究更简洁的一种即可。

2.1.4 对 NP 的两个技术性说明

回想在定义 PC （或 NP ）时，我们明确地将注意力局限在多项式界限搜索问题（或多项式长度的 NP-证据）。在另一种定义中，如果 (x, y) 是否属于 R 可在关于 x 长度的多项式时间内判定（或要求验证 y 是否为 x 属于 S 成员的 NP-证据可在 $\text{poly}(|x|)$ 时间内完成），则认为二元关系 $R \in PC$ （或 $S \in NP$ ）。事实上，这意味着在判定 y 的正确性时，不需要全部读出 y （即 y 的某个子串可以代表原始定义中有效的 y ）。

我们认为， PC （或 NP ）中的问题可在指数时间内解决（即对输入的 x ，计算时间为 $\exp(\text{poly}(|x|))$ ）。穷举搜索所有可能的解（或所有可能的 NP-证据）便可做到。因此， $NP \subseteq EXP$ ，其中 EXP 表示可在指数时间内求解的判定问题类（即对输入的 x ，计算时间为 $\exp(\text{poly}(|x|))$ ）。

2.1.5 NP 的传统定义

不幸的是，定义 2.5 并非通常使用的 NP 定义。相反，习惯上， NP 被定义为可利用一种假想的设备，称为非确定性多项式时间机器（这解释了使用 NP 这个名称的来源），判定的集合类。该类假想设备之所以吸引人，是因为它（间接地）解释了 NP-证据的定义。由于读者在阅读其他文献时可能会遇到 NP 的传统定义，所以笔者认为必须给出传统定义，并证明其与定义 2.5 等价。

定义 2.7（非确定型多项式时间图灵机）

• 非确定型图灵机的定义类似于 1.2.3.2 节，不同之处在于迁移函数将符号—状态对映射为 $\Sigma \times Q \times \{-1, 0, +1\}$ 中三元组构成的子集（而非一个三元组）。相应地，某个特定即时配置的后续配置有几种可能的选择，每一种由一个不同的三元组所确定。因此，非确定型机器对于一个（固定的）给定输入的计算可能有不同的输出。

在判定问题中，通常考虑是否存在一种计算，由给定的输入开始，停机时输出 1。称非确定型机器 M 接受 x ，如果存在 M 的一种计算，对于输入 x ，在停机时输出 1。由非确定型机器接受的集合即为由该机器接受的输入构成的集合。

• 非确定多项式时间图灵机的计算步骤数是其输入长度的多项式。习惯上, NP 被定义为由某个非确定的多项式时间图灵机接受的集合类。

我们强调定义 2.7 针对的是一种假想的计算模型。具体来说, 定义 2.7 并没有参照机器(对于一个特定输入)能得到特定输出的可能计算的数量(或比例)。^①它只涉及到(对一个特定输入)能得到某个特定输出的计算的存在性。这种可能计算的存在性(在现实生活中)尚不清楚, 因为非确定机器的模型并不是要为现实中的计算机提供一个合理模型。该模型的意义在于解释某种完全不同的事物(即优美地定义 NP 类, 同时依赖于一个事实, 即定义 2.7 与定义 2.5 的等价性)。

教学注释: 定义 2.7 是否优美属于喜好问题。毫无疑问, 许多学生会发现定义 2.7 相当混乱, 因为他们假定定义中提供了某种自然的计算模型, 然后被自己对这些模型的直觉所欺骗(当然, 在使用一个假想的计算模型时, 直觉是无关紧要的)。

注意, 与本章的其他定义不同, 定义 2.7 直接参照了一种具体的计算模型。然而, 在充分考虑了非确定的计算规则之后, 可将其扩展到其他计算模型。还要注意的, 不失一般性, 可以假设迁移函数将每个可能的符号-状态对恰好映射为两个三元组(见习题 2.4)。

定理 2.8 定义 2.5 等价于定义 2.7。即集合 S 具有 NP -证明系统当且仅当存在一个能接受 S 的非确定多项式时间机器。

证明概要: 一方面, 假设集合 S 具有 NP -证明系统, 将相应的验证程序记作 V 。考虑以下的非确定多项式时间机器 M 。对于输入 x , M 经过了 $m = \text{poly}(|x|)$ 次非确定的计算步骤, (非确定地)生成串 $y \in \{0,1\}^m$, 并模仿 $V(x,y)$ 。我们强调这些非确定的步骤可能会导致生成任意的 m 比特串 y 。回顾前面有 $x \in S$ 当且仅当存在长度至多为 $\text{poly}(|x|)$ 的串 y 使得 $V(x,y)=1$ 。这蕴含了由 M 接受的集合即为 S 。

另一方面, 假设存在一个能接受 S 的非确定多项式时间机器 M 。考虑一个确定型机器 M' , 对于输入 (x,y) , 其中 y 具有足够长度, M' 模仿 M 对输入 x 的计算, 并利用 y 来确定 M 中的非确定步骤。即 M 对输入 x 计算时, 第 i 步由 y 的第 i 比特所确定(这表示在当前步骤中, 在两种可能的移动方向中选择哪一种)。注意: $x \in S$ 当且仅当存在长度至多为 $\text{poly}(|x|)$ 的 y , 使得 $M'(x,y)=1$ 。因此由 M' 能得到 S 的 NP -证明系统。□

2.1.6 对 P 不同于 NP 的支持

直觉及概念构成了我们掌握的所有知识, 所以仅有概念而没有相应的直觉, 或仅有直觉而无概念, 都不能产生知识。

伊曼努尔—康德 (1724—1804)

康德认为, 在科学(称其为(正确的)“知识”)中, 哲学因素(称为“概念”)和经验因素(称为“直觉”)都很重要。

^① 相反, 概率机的定义中涉及到了这个数量(或等价地, 机器生成特定输出的概率, 该概率在所有可能计算中均匀取值)。因此, 如果附上合适的随机源, 则涉及到自然计算模型的概率机是可以实现的, 细节见 6.1.1 节。

普遍认为 P 不同于 NP ，即 PC 中包含一些搜索问题，是不能被高效求解的，从而不能高效判定的集合存在着 NP -证明系统。这一信条同时有哲学和经验两方面因素的支撑。

- 哲学因素。 P -vs- NP 的两种形式所涉及到的问题都有很强的概念。解（搜索）问题的概念似乎假定，至少在某些情况下（如果不是所有情况），求解要比检查给定解的正确性困难得多。这可以理解为 $PC \setminus PF \neq \emptyset$ 。类似地，证明的概念似乎假定，至少在某些情况下（如果不是普遍情况），证明在确定断言正确性方面是有用的，即在提供了证明时，验证一个断言的正确性将容易得多。这可以解释为 $P \neq NP$ ，也暗示了寻找证明要比验证其正确性难得多，这与研究者和学生的日常经验都是一致的。

- 经验因素。 NP 类（或更确切地， PC 类）中包含上千个不同问题，人们尚未找到高效的求解算法。这些问题来自于不同的学科，许多科学家和工程师团队针对这些问题做了大量研究工作。而这些本质上相互独立的研究都无法对这些问题提出高效解法，这种失败似乎很难归因于纯粹的巧合或坏运气。

在本书的剩余部分，我们将采纳 P 不同于 NP 的普遍观点。在某些地方，我们将明确地使用这个假设（或者甚至更强的假设），而在其他地方，还将提出一些结论，当且仅当 $P \neq NP$ 时这些结论才有意义（即如果 $P = NP$ ，则整个 NP -完全性理论将毫无意义）。

$P \neq NP$ 的假设其实相当直观且十分诱人。这个合理的假设至今尚未得到证实，这是复杂性理论研究中许多挫折的根源之一。然而，作者认为，这种挫折感并无道理。相反，正是由于复杂性理论处理的是一些描述起来简单但又十分难解的问题，才使得该领域的研究工作是如此的激动人心。

2.1.7 哲学思考

不重视思考的读者，可以跳过这一章。

Robert Musil, 无品质的人，第 28 章

康德清楚地指出了人类科学知识的固有局限性，他认为人类所具有的知识不能超出理解方式。所谓“理解方式”是预先确定的，它们超越了任何知识的获取，并且是获取知识的先决条件。维特根斯坦则从某种意义上精练了分析过程，他认为知识必须用某种语言表达，而后者必须被某种（合理的）意义赋予机制所支配。因此，任何可能的“意义赋予机制”的固有局限性都会限制可以（有意义地）表达的范围。

两位哲学家都提到了真实世界与人类思维之间的关系。他们想当然地认为（或假定），在具有良好形式的思维领域（即逻辑），每个正确结论都可以高效地获得（即每个正确断言都可被高效地证明）。然而，哥德尔证明了这种天真的假设是不成立的。图灵则给出了另一种类似的徒劳假设，他宣称存在着良好定义的问题，无法用良好定义的方法求解。

最后一个断言超越了上述的哲学因素：其中宣称人类能力的局限性不仅仅是由于在“世界是什么样子”和我们对其建立的模型之间存在缝隙。事实上，这一断言涉及到任意合理过程的固有局限性，甚至当将该过程应用于形式良好的信息，且有形式良好的目标时，这种局限性仍存在。实际情况是，与上述的天真假设相反，并非所有形式良好的问题都可被（高效地）解决。

$P \neq NP$ 的假设超出了上述讨论。它将讨论范围局限于“清楚的”问题，即正确解可以

被高效识别的问题。事实上，在无法高效识别正确解的问题中，存在某种虚构成分。如果回避这些虚构和/或不合理的问题， $P \neq NP$ 意味着（即使有这种局限性）存在一些问题，在无法高效求解的意义下，这些问题本质上也是不可解的。即与天真的假设相反，并非所有其解可以被高效识别的问题都可被高效求解。实际上，识别解的复杂性与求解的复杂性之间的缝隙保证了问题的概念是有意义的。

2.2 多项式时间归约

我们提出计算问题之间（多项式时间）归约的一般概念，并将“Karp-归约”作为一个重要特例，因为它可以满足许多情况下的需求（并且更方便）。归约是 NP-完全性理论的核心，2.3 节将介绍 NP-完全性理论，本节我们把重点放在归约的基本性质上，并强调两种特殊应用（即将搜索和优化问题归约为判定问题）。此外，在后一种应用中，将强调归约的一般概念（即“Cook-归约”而非“Karp-归约”）。

教学注释：我们假设许多学生都曾经听说过归约，但大部分人对于归约的基本性质了解甚少。特别是，会有人将归约与 NP-完全性理论混为一谈，实际上，在 NP-完全性理论（或相对于其他某些类的完全性）之外，归约还有许多重要的应用。另外，我们相信证明搜索问题和优化问题能归约到判定问题是十分重要的。

2.2.1 归约的一般概念

归约是调用具有“指定功能”的子程序的程序。就是说，子程序的功能被详细规定，但是没有规定其操作，并且这些子程序的运行时间是用单位代价来计数的。在研究算法时我们采用图灵机模型，类似地，在研究归约时，可以使用预言（图灵）机模型。归约可以通过利用求解一个计算问题（可以是搜索或判定问题）的预言（或子程序）调用来求解另一个（搜索或判定）问题。

2.2.1.1 归约的现有定义

一般性算法归约的概念将在 1.2.3.3 节和 1.2.3.6 节中介绍。这种归约称为图灵归约（见 1.2.3.3 节），其模型是预言机（Oracle Machine）（见 1.2.3.6 节），其中不考虑主要算法（即预言机）的时间复杂度。本节主要讨论高效的（多项式时间）归约，通常称为 Cook-归约。因此，本节涉及的预言机（见定义 1.11）其运行时间是输入的多项式。我们强调预言机的运行时间是（其自身）计算的步骤数量，而预言机对于每个询问的应答可在一步内完成。

高效归约的一个重要性质是它们能将子程序的高效实现转化为对归约于其上的计算任务的高效实现。就是说，正如我们将看到的，如果一个问题可以通过 Cook-归约转化为另一问题，那么，当后者是多项式时间可解时，前者也是。

最常见的例子是判定问题到判定问题的归约，但是我们还会考虑搜索问题到搜索问题的归约以及搜索问题到判定问题的归约。注意到当归约为判定问题时，预言机就是解该判定问题的唯一有效方式（即函数 $f: \{0,1\}^* \rightarrow \{0,1\}$ 解集合 S 的成员判定问题，如果对任意 x ，当 $x \in S$ 时 $f(x)=1$ ，否则， $f(x)=0$ ）。相反地，当归约到搜索问题时，预言机并非唯一确定，因为可能存在许多不同的有效解法（即函数 $f: \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ 解 R 的搜索问题，如果对任意

$(x, y) \in R$, 当 $x \in S_R$ 时, $f(x) \in R(x)$, 否则, $f(x) = \perp$ 。下面的定义中包含了上述两种情况。

定义 2.9 (Cook-归约) 如果存在多项式时间预言机 M , 使得对任意一个求解问题 Π' 的函数 f , M^f 可以解问题 Π , 则称 Π 可以 Cook 归约为 Π' , 其中 M^f 表示 M 在可以调用预言 f 时对于输入 x 的输出。

注意到 Π (或 Π') 既可以是搜索问题也可以是判定问题 (或者甚至是类型未定义的问题)。在这里读者应该能证明如果问题 Π 可以 Cook-归约为 Π' , 则当 Π' 可以在多项式时间内求解时, Π 也能 (见习题 2.5 中关于 Cook-归约的其他性质)。

观察到定理 2.6 证明中的第二部分实际上就是一个 Cook-归约, 将 PC 中的搜索问题 R 归约为 NP 类关系集 $S'_R = \{(x, y') : \exists y'' \text{ 使得 } (x, y'y'') \in R\}$ 的判定问题。因此, 该证明过程证实了如下结论。

定理 2.10 PC 中任意一个搜索问题可以 Cook-归约为 NP 中某个判定问题。

我们将在 2.2.3 节中看到搜索问题与判定问题间存在更紧密的关系, 即在某些情况下 R 可以归约为 $S_R = \{x : \exists y, \text{ 使得 } (x, y) \in R\}$ 而非 S'_R 。

2.2.1.2 特例

Karp-归约是 (由判定问题到判定问题) 归约的一种特例。具体来说, 对于判定问题 S 和 S' , 称 S 可以 Karp-归约为 S' , 如果存在以如下方式的 S 到 S' 的归约: 对于输入 x (S 的一个实例), 归约时计算出 x' , 将 x' 作为对于预言 S' 的询问 (即对于输入 x' 调用子程序 S'), 并将后者的应答作为归约的计算结果。这个归约通常用多项式时间内计算的 x 到 x' 的映射来表示, 从而 Karp-归约的标准定义如下所述。

定义 2.11 (Karp-归约) 一个多项式时间内计算的函数 f 称为 S 到 S' 的 Karp-归约, 如果对任意 x , $x \in S$ 当且仅当 $f(x) \in S'$ 。

因而, 从语法角度讲, Karp-归约不是 Cook-归约, 但它可以平凡地诱导出一种 Cook-归约 (即对于输入 x , 预言机询问 $f(x)$, 并返回应答)。一种不十分准确但实际上有道理的说法是: Karp-归约是 Cook-归约的特例。当然, Karp-归约是 Cook-归约的一种受限情况。尽管如此, 这种受限情况可以满足许多应用 (如特别是用于 NP-完全性理论当中 (其研究进展使用判定问题时)), 但是不能将搜索问题归约为判定问题。另外, 虽然任意一个判定问题都可以 Cook-归约为其补问题, 但某些判定问题不能 Karp-归约为其补问题 (见习题 2.7 和习题 2.33)。

在此说明, Karp-归约可以 (应该) 被扩展到搜索问题到搜索问题的归约。这种扩展后的 Karp-归约将搜索问题 R 归约为搜索问题 R' , 其工作方式如下: 对于输入 x (R 的一个实例), 计算出 x' , 向预言 R' 询问 x' (即对于输入 x' 调用子程序 R'), 得到应答 y' 满足 $(x', y') \in R'$, 再利用 y' 计算出对于输入 x 的解 y (即 $y \in R(x)$)。因此, 这种归约可以表示为两个多项式时间映射, 即 f 和 g , 使得对 $f(x)$ 的任意解 y' (即满足 $(f(x), y') \in R'$ 的 y'), 有 $(x, g(x, y')) \in R$ (实际上, 一般来说, 与判定问题情况不同, 这时的归约不能仅仅返回 y' 作为 x 的解)。上述的扩展称为 Levin-归约, 并且与 Karp-归约类似的是, Levin-归约也常用两个多项式时间映射表示 (即 x 到 x' , 以及 (x, y') 到 y 的映射)。

定义 2.12 (Levin 归约) 一对多项式时间计算的映射 f 和 g 称为 R 到 R' 的 Levin-归约,

如果 f 是 $S_R = \{x: \exists y, \text{使得}(x, y) \in R\}$ 到 $S_{R'} = \{x': \exists y', \text{使得}(x', y') \in R'\}$ 的 Karp-归约, 并且对任意 $x \in S_R$ 和 $y' \in R'(f(x))$, 有 $(x, g(x, y')) \in R$, 其中 $R'(x') = \{y': (x', y') \in R'\}$ 。

实际上, 函数 f 表示解的存在性, 即对任意 x , $R(x) \neq \emptyset$ 当且仅当 $R'(f(x)) \neq \emptyset$ 。至于第二个函数 (即 g), 则是将归约后的实例 $f(x)$ 的解 y' 映射为原始实例 x 的解 (这种映射可能还依赖于 x)。注意到人们还可能很自然地考虑第三个函数, 即把 R 的解映射为 R' 的解 (见习题 2.28)。

2.2.1.3 术语及简要讨论

在以下的章节中, 如果归约类型被忽略, 则认为是 Cook-归约。下面介绍两个经常用到的传统术语。

- 如果两个问题可以相互归约, 则称它们在计算上等价 (Computationally Equivalent)。这意味着这两个问题难度相同。注意: 计算上等价的问题可能不属于同一个复杂性类。

例如, 2.2.3 节将指出, 存在许多 $R \in \mathcal{PC}$ 使得 R 的搜索问题与判定问题 $S_R = \{x: \exists y, \text{使得}(x, y) \in R\}$ 在计算上等价, 虽然 (甚至仅从语言层面讲) 这两个问题不属于同一类 (即 $R \in \mathcal{PC}$ 而 $S_R \in \mathcal{NP}$)。还有, 每个判定问题在计算上等价于其补问题, 虽然这两个问题也可能不属于同一类 (见 2.4.3 节)。

- 称问题类 C 可以归约为一个问题 Π' , 如果 C 中每个问题都能归约到 Π' 。称问题类 C 可以归约为问题类 C' , 如果对任意 $\Pi \in C$, 存在 $\Pi' \in C'$ 使得 Π 可以归约为 Π' 。

例如, 定理 2.10 表明 $\mathcal{PC}$ 可归约为 $\mathcal{NP}$ 。

Cook-归约使我们在判定问题与其他计算问题间建立起重要的联系。特别地, 如 2.2.2 节中将指出, 一类优化问题可以归约到 $\mathcal{NP}$ 。另外, $\mathcal{PC}$ 也可以归约为 $\mathcal{NP}$ (见定理 2.10), 还有在 2.2.3 节中将看到, $\mathcal{PC}$ 中许多搜索问题可以归约为 $\mathcal{NP}$ 中相应的判定问题 (而不是仅仅归约为 $\mathcal{NP}$ 中某个问题)。所有这些结论的推导中使用的都是 (且必须是) Cook-归约。

2.2.2 优化问题到搜索问题的归约

许多搜索问题涉及到与每个问题实例相关的解集, 进而为不同的解分配不同的“值”(或“开销”)。此时, 我们会对寻找超过某个阈值 (或低于某个门限开销) 的解感兴趣。换句话说, 人们可能会寻找具有最大值 (或最小开销) 的解。为简单起见, 我们只考虑要使解取最大值的情况。这里仍存在两个不同的目标 (即超过某个阈值和优化), 并引出相应于某个关系 R 的两个不同的 (辅助的) 搜索问题。特别地, 对于二元关系 R 和赋值函数 $f: \{0,1\}^* \times \{0,1\}^* \rightarrow R$, 考虑如下两个搜索问题。

(1) 超过一个阈值。给定一对 (x, v) , 求 $y \in R(x)$ 使得 $f(x, y) \geq v$, 其中 $R(x) = \{y: (x, y) \in R\}$ 。实际上, 指的是与关系

$$R_f \stackrel{\text{def}}{=} \{(\langle x, v \rangle, y): (x, y) \in R \wedge f(x, y) \geq v\} \quad (2.1)$$

相对应的搜索问题, 其中 $\langle x, v \rangle$ 表示对 (x, v) 编码后的串。

(2) 最大化。给定 x , 求 $y \in R(x)$ 使得 $f(x, y) = v_x$, 其中 v_x 表示对所有 $y' \in R(x)$, $f(x, y')$ 的最大值。实际上, 指的是与关系

$$R'_f \stackrel{\text{def}}{=} \{(x, y) \in R: f(x, y) = \max_{y' \in R(x)} \{f(x, y')\}\} \quad (2.2)$$

相对应的搜索问题。

赋值函数的例子包括图中团的规模、网络中的流量等。相应的计算任务可能是在给定图中寻找规模超过给定阈值的团,或寻找具有最大规模的团。请注意,在这些例子中,“基本的”搜索问题(即关系 R)是很容易解决的,困难来自于对解的赋值的辅助条件(表示为 R_f 和 R'_f)。

实际上,可以将 R 平凡化(即对任意 x , 令 $R(x) = \{0,1\}^{\text{poly}(|x|)}$),并通过函数 f 来实现所有必要的结构(见习题 2.8)。

我们只讨论 f 为多项式时间的情况,这意味着 $f(x,y)$ 可以表示为一个有理数,其长度是 $|x|+|y|$ 长度的多项式。下面将证明,在这种情况下,上述两个搜索问题(即 R_f 与 R'_f)在计算上等价。

定理 2.13 对任意多项式时间计算的函数 $f: \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbf{R}$ 及多项式界关系 R , 令 R_f 和 R'_f 分别如式 (2.1) 和式 (2.2) 所定义, 则关于 R_f 和 R'_f 的搜索问题在计算上等价。

注意: 对于 $R \in PC$ 及多项式时间计算的函数 f , 有 $R_f \in PC$ 。结合定理 2.10 和定理 2.13, 可以得出 R_f 和 R'_f 都能归约到 NP 的结论。然而, 要注意到, 即使在这种情况下, $R'_f \in PC$ 也不一定成立。见定理 2.13 证明之后的进一步讨论。

证明: 通过(对给定实例)寻找最优解, 并与给定阈值相比较, 可将搜索问题 R_f 归约为搜索问题 R'_f 。这就是说, 我们可以构造一个预言机, 通过对 R'_f 的一次询问便能解 R_f 。具体地, 对于输入 (x,v) , 机器(向解 R'_f 的算法)发出询问 x , 并得到优化的解 y (或表示 $R(x) = \emptyset$ 的符号“ $\perp$ ”), 然后计算 $f(x,y)$, 如果 $f(x,y) \geq v$, 则返回 y ; 否则(即 $y = \perp$ 或者 $f(x,y) < v$), 返回表示 $R_f(x,v) = \emptyset$ 的指示。

另一方面, 可将搜索问题 R'_f 归约为搜索问题 R_f , 方法是首先找到最优值 $v_x = \max_{y \in R(x)} \{f(x,y)\}$ (对所有可能的值进行二分法查找), 然后求关于 v_x 的解。这两个步骤中都需要调用预言 R_f 。为简单起见, 假设总是给 f 分配正值, 并令 $l = \text{poly}(|x|)$, 使得对任意 $y \in R(x)$, 有 $f(x,y) \leq 2^l - 1$, 则对于输入 x , 首先通过对 $\langle x,v \rangle$ 调用预言而找到 $v_x = \max \{f(x,y) : y \in R(x)\}$, 这里的要点是, $v_x < v$ 当且仅当 $R_f(\langle x,v \rangle) = \emptyset$, 而这又可由预言(对于查询 $\langle x,v \rangle$)的应答“ $\perp$ ”来指出。经过 l 次查询后, 便可确定 v_x (见习题 2.9)。注意: 当 $R(x) = \emptyset$ 时, 所有的应答都暗示了 $R_f(\langle x,v \rangle) = \emptyset$, 此时, 我们认为 $v_x = 0$ 。最后, 提出查询 $\langle x, v_x \rangle$, 并当接收到预言的应答(为 $y \in R(x)$, 且当 $v_x > 0$ 时, $f(x,y) = v_x$, 否则, 指出 $R(x) = \emptyset$)时停止且返回该应答。 ■

思考: 注意前一部分利用了 f 可在多项式时间计算的假设, 而后一部分只利用了一个事实, 即最优值属于某个指数规模的有限空间, 并且能被“高效地找到”。在第一部分证明中利用了 Levin-归约, 这在后一部分中似乎是不可能的(一般而言)。

R_f 和 R'_f 的复杂性

我们关注一种自然的情况, 即 $R \in PC$ 且 f 可在多项式时间计算。此时, 定理 2.13 蕴含了 R_f 与 R'_f 在计算上等价。然而, 通过进一步思考会发现, $R_f \in PC$ 总成立, 而 $R'_f \in PC$ 并不总是成立。从而, (对于给定实例)求超过给定阈值的解属于 PC 类, 而求最优解不一定属于 PC 类。例如, 在给定图 G 中求具有规模 K 的团是一个 PC 问题, 而求具有最大规模的团这

一问题是否属于 PC (虽然它可以 Cook-归约到 PC 类) 尚且未知 (而且似乎没有这种可能性)。事实上, 可以归约为 PC 类的问题构成了一类很有意思的问题类 (见 3.2.1 节末尾的讨论)。显然, 对任意的 $R \in PC$ 及多项式时间计算的 f , 这个类中包含了 R'_f 。

2.2.3 搜索问题的自归约性

本节给出的结论将进一步表明重点讨论判定问题是合理的。不严格地讲, 这些结论说明了对于许多自然关系 R , R 的搜索问题是否可被高效解决 (即属于 PF 类) 取决于 “ R 中蕴含的判定问题” (即 $S_R = \{x: \exists y \text{ 使得 } (x, y) \in R\}$) 是否可以高效解决 (即属于 P 类)。注意: 后一个判定问题可以容易地归约为 R 的搜索问题, 所以我们只考虑另一方向, 即只关心那些与其相关的搜索问题可以归约为 S_R 的判定问题的关系 R 。

教学注释: 这里使用的自归约这个术语有别于其传统含义。习惯上, 一个判定问题是 (下行) 自归约的, 如果它可以 Cook-归约为其自身, 且归约中是对于输入 x , 只询问比 x 小的值 (根据某种恰当的对于串长度的测度)。在某些自然的约束条件下 (如归约中采用预言应答的析取式) 这种归约可以诱导出由搜索问题到判定问题的归约 (如本章正文中所讨论的)。详见习题 2.13。

定义 2.14 (搜索问题中蕴含的判定问题及自归约性) 关系 R 的搜索问题中蕴含的判定问题是指判定集合 $S_R = \{x: R(x) \neq \emptyset\}$ 的成员问题, 其中 $R(x) = \{y: (x, y) \in R\}$ 。如果 R 的搜索问题可以归约为 S_R 的判定问题, 则称其为自归约的。

注意: R 的搜索问题以及 S_R 的成员判定问题指向相同的实例: 搜索问题要求找到一个充分的解 (即对给定的 x , 求 $y \in R(x)$), 而判定问题是要判断这样的解是否存在 (即对给定的 x 判断 $R(x)$ 是否为空)。因此, S_R 实际上就是 “ R 中所蕴含的判定问题”, 因为在解 R 的搜索问题时已经间接地解决了判定问题 S_R 。事实上, 对任意 R , S_R 的判定问题易于归约为 R 的搜索问题 (并且如果 R 属于 PC 类, 则 S_R 属于 NP 类)。^① 由此可以得到, 如果搜索问题 R 是自归约的, 则它与判定问题 S_R 在计算上等价。

注意到自归约似乎在本质上属于归约的一般概念 (即 Cook-归约)。这不仅仅是出于语法上的考虑, 而更多的是由于以下的内在原因。任意一个判定问题的预言机在每次调用时返回一个比特, 而 PC 中搜索问题的不可解性一定是因为缺乏比 “1 比特信息” 更多的信息 (见习题 2.10)。

我们将会看到许多搜索问题都具有自归约性 (包括所有的 NP -完全搜索问题)。这在更强的意义上解释了 2.1.3 节中所建立起的关于判定问题与搜索问题的相关性。2.1.3 节证明了搜索问题类 PC (或 PF) 的命运取决于判定问题类 NP (或 P) 的命运。这里将证明, 对于 PC 中许多搜索问题 (即那些具有自归约性的问题) 而言, 一个这样的问题 R (或 PF) 的命运取决于判定问题 S_R (或 P) 的命运, 其中 S_R 为 R 中所蕴含的判定问题。

2.2.3.1 例子

现在介绍几个具有自归约性的搜索问题。我们从 SAT (见附录 G.2) 开始, 介绍可满足的布尔表达式 (在 CNF 中) 集合, 并考虑这样一个搜索问题: 给定一个表达式, 求可满足的

^① 例如, 归约时调用搜索预言, 当且仅当预言返回某个串 (而非表示 “无解” 的符号) 时应答为 1。

一种真值赋值。相应的二元关系记作 R_{SAT} ，它表示如果 τ 是表达式 Φ 的一种可满足的赋值，则 $(\Phi, \tau) \in R_{\text{SAT}}$ 。蕴含于 R_{SAT} 中的判定问题实际上就是 SAT。注意到 R_{SAT} 属于 $\mathcal{PC}$ （即它是多项式界的，且易判定 (Φ, τ) 是否为 R_{SAT} 中成员（直接计算布尔表达式的值即可））。

命题 2.15 (R_{SAT} 是自归约的) 搜索问题 R_{SAT} 可以归约为 SAT。

从而，搜索问题 R_{SAT} 与 SAT 中成员的判定问题在计算上等价。因此，我们在研究 SAT 的复杂性的同时，也研究了搜索问题 R_{SAT} 的复杂性。

证明：构造一种通过调用预言 SAT 来解 R_{SAT} 的预言机。给定一个表达式 Φ ，用如下方法找到 Φ 的可满足的赋值（如果这样的赋值存在）。首先，用 Φ 自身来询问 SAT，如果无解，则预言应答为 0（表示 $\Phi \notin \text{SAT}$ ）；否则，用 τ 表示 Φ 的可满足性赋值的前缀，并将其初始化为空串。下面进行迭代，每次迭代中为 τ 增加 1 比特，方法如下：首先通过对 Φ 中的 $|\tau|+1$ 个变量赋值为 τ_0 而得到一个新的表达式，记作 Φ' ，然后向 SAT 询问 Φ' （即询问 τ_0 是否为 Φ 的一个可满足性赋值的前缀）。如果得到肯定回答，则令 $\tau \leftarrow \tau_0$ ，否则 $\tau \leftarrow \tau_1$ 。这个过程利用了一个事实，即如果 τ 是 Φ 的一个可满足性赋值的前缀而 τ_0 不是，则 τ_1 必为 Φ 的一个可满足性赋值的前缀。

要强调在上述描述中一个含义不太清晰的重要概念。上述的表达式 Φ' 是通过用某些常量来代替变量而得到的，这意味着 Φ' 本身既含布尔变量也含布尔常量。然而在 SAT 的标准定义中，不允许实例中出现布尔常量。^①尽管如此， Φ' 仍可以被简化为不含布尔常量的表达式。这个简化过程通过使用布尔运算规则来实现，即常量“假”（False）可以从所有子句中省略掉，但是如果一个子句只包含常量“假”，则整个布尔表达式可以化简为“假”。类似地，如果常量“真”出现在某个子句中，则整个子句可被省略，但是如果所有的子句都被省略了，则整个布尔表达式可化简为“真”。当然，如果通过化简过程得到一个布尔常量，则可以跳过这个询问；否则，只需使用化简后的 Φ' 来询问。 ■

其他例子

类似于命题 2.15 证明中所使用的归约也可用于其他搜索问题中（而不仅仅是 NP-完全问题）。例如，求给定图的 3-着色方案，以及求一对给定图之间的同构映射（这里众所周知前一个问题为 NP-完全问题，而第二个问题则被认为不是 NP-完全问题）。在两个例子当中，搜索问题到判定问题的归约都涉及到通过适当的询问来延展合法解的一个前缀，而延展的内容则取决于询问结果。然而，要注意的是，在这些例子中，常量的去除是十分复杂的。例如，在图的 3-着色问题（或图的同构问题）中，我们需要对给定图的一部分着色（或找出一对给定图的部分同构），分别见习题 2.11 和习题 2.12。

思考

命题 2.15 的证明（以及类似结论的证明）中包含两个要点：

(1) 对 $\mathcal{PC}$ 中的任意关系 R ， R 的搜索问题可归约为 S'_R 的判定问题，其中 $S'_R = \{(x, y') : \exists y'' \text{ 使得 } (x, y'y'') \in R\}$ 。定理 2.6 的证明中明确地构造了这种归约，而在命题 2.14 的证明中则不太明确。

(2) 对于特殊的 $R \in \mathcal{PC}$ （如 S_{SAT} ）， S'_R 的成员判定问题可归约为 S_R 的成员判定问题，其

^① 虽然这一问题在目前环境下似乎需要相当多的技巧（因为它仅仅归结为在 SAT 的定义中是否允许其实例中包含布尔常量），但在其他情况下则不需要那么多技巧（见习题 2.11 和习题 2.12）。

中 $S_R = \{x: \exists y \text{ 使得 } (x, y) \in R\}$ 。这就是为什么在证明中使用 SAT 的特殊结构的原因, 因为这样可以构造由 S'_R 实例到 S_R 实例的直接且自然的变换。

(我们指出如果 S_R 是 NP-完全的, 那么, 由于 S'_R 总是属于 $\mathcal{NP}$ 类, 从而仅仅根据 S_R 是 NP-完全的, S'_R 便可归约为 S_R 。以下给出进一步的讨论。)

对任意的 $R \in \mathcal{PC}$, S'_R 的成员判定不一定能归约为 S_R 的成员判定。另外, S'_R 的成员判定也不一定能归约为搜索问题 R (见习题 2.14、习题 2.15 和习题 2.16)。

通常自归约性是搜索问题的性质, 而搜索问题中蕴含的判定问题则不具有该性质。另外, 在合理的假设下 (如假设整数分解问题是难解的), 存在关系 $R_1, R_2 \in \mathcal{PC}$, 蕴含着同一个判定问题 (即 $\{x: R_1(x) \neq \Phi\} = \{x: R_2(x) \neq \Phi\}$), 且 R_1 是自归约的, 而 R_2 不是 (见习题 2.17)。

然而, 对于许多判定问题, 这种现象不会发生, 即对于许多自然的 NP-类判定问题 S 而言, 任意一个与 S 相关的 NP-证据关系 (即 $R \in \mathcal{PC}$ 使得 $\{x: R(x) \neq \Phi\} = S$) 都是自归约的。具体可参考命题 2.15 之后的例子, 以及 2.2.3.2 节中的进一步讨论。

2.2.3.2 NP-完全问题的自归约性

教学注释: 在以下的进一步讨论中, 我们假设学生知道 NP-完全性, 实际上, 这里只需要了解 NP-完全性的定义 (即集合 S 是 $\mathcal{NP}$ -完全的, 如果 $S \in \mathcal{NP}$ 且 $\mathcal{NP}$ 中任意一集合可归约为 S) 即可。然而, 教师可将下述的前沿性讨论推迟至 2.3.1 节中 (或更靠后的章节中) 讲述。

一般情况下, 自归约性是 R 的搜索问题的性质, 搜索问题中蕴含的判定问题 (即 $S_R = \{x: R(x) \neq \Phi\}$) 则不具有该性质。然而, 在 NP-完全问题的一些特例中, 与 (NP-完全) 判定问题相关联的任意证据关系也具有自归约性。这就是说, 与任意一个 NP-完全判定问题相关的求 NP-证据的搜索问题都是自归约的。

定理 2.16 对任意使得 S_R 为 NP-完全问题的 $R \in \mathcal{PC}$, R 的搜索问题可以归约为 S_R 的成员判定。

在许多情况下, 与命题 2.15 的证明相似, 搜索问题到相应判定问题的归约是非常自然的。以下证明中给出了一种新的归约方法 (在某些情况下可能不那么 “自然”)。

证明: 为了将搜索问题 R 归约为判定问题 S_R , 我们构造如下两个归约:

(1) 将搜索问题 R 归约为 $S'_R = \{(x, y'): \exists y'' \text{ 使得 } (x, y'y'') \in R\}$ 的成员判定。

上一段中说明了 (标题为 “思考” 的一段), 命题 2.15 的证明暗示了这种归约 (而在定理 2.6 的证明中则明确定义了这种归约)。

(2) 将 S'_R 归约为 S_R 。

这个归约建立在假设 S_R 是 $\mathcal{NP}$ -完全问题而 $S'_R \in \mathcal{NP}$ 的基础上 (注意: 这里并没有假设这种归约是 Karp 归约, 另外, 它还可以是一种 “不自然的” 归约)。

定理得证。 ■

2.2.4 总结及一般性观点

回顾我们曾提出 (多项式时间) 归约是使用详细定义了功能的子程序的 (高效) 算法。也就是说, 问题 Π 到问题 Π' 的高效归约是一个解问题 Π 的高效算法, 算法在运行时能调用任意一个解 Π' 的算法作为子程序。这种表示符合归约的 “正常” (“积极”) 应用; 即将这种归

约与(解 Π 的)子程序的高效实现相结合,可以得到解 Π 的高效算法。注意:在 Π 与 Π' 间存在多项式时间归约的实际含义要比上述的应用更为丰富。例如,即使是在这种归约中使用解 Π' 的效率较低的算法,也可为 Π 的求解提供帮助,也就是说,如果 Π' 可在时间 t' 内求解,则 Π 可在时间 t 内求解,且 $t(n) = \text{poly}(n) \cdot t'(\text{poly}(n))$ (例如,如果 $t'(n) = n^{\log_2 n}$,则 $t(n) = n^{O(\log n)}$)。因此, Π 与 Π' 之间多项式时间归约的存在性,提供了 Π 的时间复杂度在用 Π' 的时间复杂度表示时的上限。

注意到 Π 与 Π' 的时间复杂度间可以建立更紧密的联系,而无论归约是否满足附加性质。例如,假设 Π 可以多项式时间内归约到 Π' ,且归约时发出的询问具有线性长度(即对输入 x ,每个询问的长度为 $O(|x|)$)。从而,如果 Π' 可在时间 t' 内求解,则 Π 可在时间 t 内求解,且满足 $t(n) = \text{poly}(n) \cdot t'(O(n))$ (例如,当 $t'(n) = 2^{\sqrt{n}}$ 时, $t(n) = 2^{O(\sqrt{n})}$)。进一步地,要注意对于归约的其他复杂性测度(如空间复杂度)求上限时,必须考虑到在问题间的相对复杂度之间建立联系,见5.2.2节。

与上述多项式时间归约的“正面”应用相反,NP-完全性理论(见2.3节)是由于这种归约的“负面”应用而著名的,详述如下。问题 Π 可在多项式时间内归约到 Π' ,这一事实意味着如果 Π' 是可解的,则 Π 也是可解的。直接的“正面”应用从假设 Π' 可解开始,最终推导出 Π 也是可解的。相反,“负面”应用利用了完全相反的方式:从假设对 Π 求解不可行开始,推导出 Π' 也不可解。

2.3 NP-完全性

鉴于P-vs-NP问题不可解,面对NP中的困难问题H时,我们不能期望(无条件地)证明H不属于P。最好的情况是在 $\text{NP} \neq \text{P}$ 的假设下证明H不属于P。相反,如果能证明H属于P,则NP中任何问题都属于P(即 $\text{P} = \text{NP}$)。证明这个断言一种可能的方法是证明NP中任意问题都能多项式归约到H。这就是NP-完全性理论的精髓。

教学注释:有些学生曾经听说过NP-完全性,但我们怀疑许多人并未理解其概念要点。特别是他们可能会忽视一点,即仅仅是NP-完全问题的存在性也足以令人惊奇(更不要说许多如此常见的问题,如SAT,竟然属于这一类)。我们认为之所以会出现这种情况,是由于许多教科书中在定义了NP-完全性之后马上给出了Cook定理的证明细节。

2.3.1 定义

NP-完全性的标准定义使用了判定问题。下面我们还会给出NP-完全搜索问题(或PC-完全问题)的定义。在两种情况下,问题 Π 的NP-完全性都包含两个条件:

- (1) Π 属于相应的类(即 Π 在NP中或PC中,这取决于 Π 是判定问题还是搜索问题)。
- (2) 该类中的所有问题都能归约到 Π ,这个条件称为NP-困难性。

虽然NP-困难性的良好定义允许使用任意的Cook归约,但结果是利用Karp-归约(或Levin-归约)也足以证明任意一个自然的NP-完全判定(搜索)问题的NP-完全性。因此,NP-完全性通常利用如下有限制的多项式时间归约概念来定义。

定义 2.17 (判定问题的 NP-完全性, 受限概念) 集合 S 是 $\mathcal{NP}$ -完全的, 如果 S 属于 $\mathcal{NP}$, 且 $\mathcal{NP}$ 中任意集合可以 Karp-归约到 S 。

一个集合是 $\mathcal{NP}$ -困难的, 如果 $\mathcal{NP}$ 中任意集合可以 Karp-归约到这个集合。事实上, 这里没有必要一定要用 Karp-归约 (而非任意的 Cook 归约), 除非上述的受限概念足以适用于所有的 NP-完全问题并且易于处理。关于搜索问题有一个类似定义。

定义 2.18 (NP-完全的搜索问题, 受限概念) 二元关系 R 是 $\mathcal{PC}$ -完全的, 如果 R 属于 $\mathcal{PC}$ 且 $\mathcal{PC}$ 中任意关系可以 Levin-归约到 R 。

最后, 我们有时会滥用一下术语, 将搜索问题也当作 NP-完全的 (而非 $\mathcal{PC}$ -完全)。类似地, 如果 $\mathcal{PC}$ 中任意关系可以归约到该问题, 我们会说一个搜索问题是 NP-困难的 (而非 $\mathcal{PC}$ -困难的)。

在此强调一个事实, 就是仅仅给出一种性质的定义 (即 NP-完全性), 并不意味着存在满足该性质的对象。事实上, NP-完全问题的存在性是非常重要的。这样的问题被看作具有“通用性”, 因为对于它们的成功解决意味着其他 (合理) 问题 (即 NP 问题) 的解决。

2.3.2 NP-完全问题的存在性

NP-完全问题的存在性这个问题本身很重要, 但更重要的是是否存在“自然”的 NP-完全问题, 注意不要混淆这两者。以下的证明解决了第一个问题, 并且重点讨论了 NP-完全性的本质, 而不是更复杂的技术细节。NP-完全性的本质是某个计算问题可以对一大类看上去似乎毫无关系的问题进行“高效编码”。

定理 2.19 存在 NP-完全的关系及集合。

证明: 证明 (以及其他 NP-完全性的证明) 基于这样一个观察, 即 $\mathcal{NP}$ 中某些判定问题 (或 $\mathcal{PC}$ 中某些搜索问题) “足以”对 $\mathcal{NP}$ 中所有的判定问题 (或 $\mathcal{PC}$ 中所有搜索问题) 进行编码。这个事实对于“一般的”判定问题和搜索问题而言是非常明显的, 分别记作 S_u 和 R_u (定义稍后给出), 我们利用这些问题来得到本定理一种最简单的证明。

考虑如下定义的关系 R_u 及其中蕴含的判定问题 S_u (即 $S_u = \{\bar{x} : \exists y \text{ 使得 } (\bar{x}, y) \in R_u\}$)。

两个问题涉及到同一种实例, 具有形式 $\bar{x} = \langle M, x, l' \rangle$, 其中 M 是关于 (确定型) 图灵机的描述, x 为字符串, l' 为正整数。为了适应各种复杂性量度, l' 以整数形式给出 (而非二进制), 因为这些量度依赖于问题实例的长度, 通常是 l' 的多项式 (而非 l' 的对数的多项式)。

定义

关系 R_u 由满足下述条件的对 $(\langle M, x, l' \rangle, y)$ 组成: M 在 l' 步内接受输入对 (x, y) , 其中 $|y| \leq l'$ 。^① 相应的集合 $S_u \stackrel{\text{def}}{=} \{\bar{x} : \exists y, \text{ 使得 } (\bar{x}, y) \in R_u\}$ 由一些三元组 $\langle M, x, l' \rangle$ 组成, 其中机器 M 在 l' 步内接受具有形式 $(x, \cdot)$ 的输入。

显然, R_u 属于 $\mathcal{PC}$ 而 S_u 属于 $\mathcal{NP}$ 。事实上, R_u 可以被通用的图灵机识别, 该机器对于输入 $(\langle M, x, l' \rangle, y)$, 模仿 M 对 (x, y) 的 (l' 步) 计算 (类似地, 可以得到 $S_u \in \mathcal{NP}$)。这里的下

① 可以不要求 $|y| \leq l'$, 而只要求 M 是“规范的”, 即在停机前能读到所有的输入。

标 u 表示“通用”(Universal)(即通用机), 而证明过程可以扩展到任意合理的计算模型(其中包含足够的通用机)。

现在证明 R_u 和 S_u 在充分意义上是 NP-困难的(即 R_u 为 PC-困难而 S_u 为 NP-困难)。首先证明 NP 中任意集合可以 Karp-归约为 S_u 。设 S 为 NP 中一个集合, 现在通过 R 来定义 S 的证据关系, 即 $R \in PC$ 且 $x \in S$, 当且仅当存在 y , 使得 $(x, y) \in R$ 。令 p_R 为 R 的解长度的多项式界(即对任意 $(x, y) \in R$, $|y| \leq p_R(|x|)$), 令 M_R 为判定 R 中成员(对给定的对 (x, y)) 的多项式时间机器, 并令 t_R 为关于 M_R 运行时间的多项式上界, 则可定义一个 Karp-归约将实例 x (S 的实例) 映射为实例 $\langle M_R, x, 1^{t_R(|x|+p_R(|x|))} \rangle$ (S_u 的实例), 即

$$x \mapsto f(x) \stackrel{\text{def}}{=} \langle M_R, x, 1^{t_R(|x|+p_R(|x|))} \rangle \quad (2.3)$$

注意: 这个映射可在多项式时间内计算, 并且 $x \in S$ 当且仅当 $f(x) = \langle M_R, x, 1^{t_R(|x|+p_R(|x|))} \rangle \in S_u$ 。

证明细节如下:

首先, 注意到映射 f 不依赖于 S , 所以它有可能依赖于固定的 M_R 、 p_R 和 t_R (这些又取决于 S)。因此, 对于输入 x 计算函数 f 时需要写下串 M_R , 复制 x , 并写下一串 1, 其数量为 $|x|$ 的某个固定的多项式。由此得到 f 可在多项式时间内计算。其次, 注意到 $x \in S$ 当且仅当存在 y 使得 $|y| \leq p_R(|x|)$ 且 $(x, y) \in R$ 。由于 M_R 在 $t_R(|x|+|y|)$ 步内接受 $(x, y) \in R$, 从而有 $x \in S$ 当且仅当存在 y 使得 $|y| \leq p_R(|x|)$ 且 M_R 在 $t_R(|x|+|y|)$ 步内接受 (x, y) 。这样就证明了 $x \in S$ 当且仅当 $f(x) \in S_u$ 。

现在证明搜索版本。为了将任意的搜索问题 $R \in PC$ 归约到搜索问题 R_u , 我们使用与判定版本基本相同的归约。对于一个输入实例 x (R 的实例), 向搜索问题 R_u 询问 $\langle M_R, x, 1^{t_R(|x|+p_R(|x|))} \rangle$, 并返回后者的应答。注意到, 如果 $x \notin S$, 则应答为“无解”, 而对任意的 x 和 y , $(x, y) \in R$ 当且仅当 $\langle M_R, x, 1^{t_R(|x|+p_R(|x|))} \rangle, y \in R_u$ 。因此, R 到 R_u 的 Levin-归约中包含一对函数 (f, g) , 其中 f 为判定版本中的 Karp-归约, 而 $g(x, y) = y$ 。注意: 实际上对任意 $(f(x), y) \in R_u$, 有 $(x, g(x, y)) = (x, y) \in R$ 。

前沿性讨论

注意到在 R_u 的定义中 1^t 的作用是令 R_u 属于 PC。相反, 考虑关系 R'_u , 其中包含对 $\langle \langle M, x, t \rangle, y \rangle$, 使得 M 在 t 步内接受 xy 。实际上, 两者的差别在于, R_u 中的时间上界 t 是一元的, 而在 R'_u 中是二元的。所以, 在 4.2.1.2 节中将会很明显地看到, R'_u 的成员判定不可能在多项式时间内完成。更进一步, 我们注意到如果从问题实例中省略 t , 将会得到一个根本不可能解决的搜索问题, 即得到关系 $R_H \stackrel{\text{def}}{=} \{ \langle \langle M, x \rangle, y \rangle : M(xy) = 1 \}$ (这个关系与停机问题相关)。实际上, 任何一个关系(特别是 PC 中的任意关系)的搜索问题可以 Karp-归约到搜索问题 R_H , 但是后者根本不可能解决(即不存在算法, 对任意的输入都停机, 且对于输入 $\bar{x} = \langle M, x \rangle$, 输出 y 使得 $(\bar{x}, y) \in R_H$ 当且仅当这样的 y 存在)。

有界停机与不停机

注意到在定理 2.19 的证明中出现的 NP-完全问题与以下两个问题相关, 即有界停机 (Bounded Halting) 和有界不停机 (Bounded Non-Halting)。在任意一种编程语言下, 这两个问题的实例中都包含程序 π 和时间上界 t (一元表示)。有界停机 (或有界不停机) 问题的判定版本是指判定是否存在一种 (长度不超过 t 的) 输入使得程序 π 在 t 步内停机 (或在 t 步内不停机), 而搜索版本是求出这样一种输入。

有界不停机问题的判定版本与程序验证 (Program Verification) 领域一个基本的计算问题相关, 具体来说, 就是判定一个给定程序在给定时间上界内对所有具有给定长度的输入是否停机的问题。^①这里提到有界停机问题是因为常在一些文献中见到这个问题, 而我们相信有界不停机问题与程序验证的关系更为密切 (因为人们感兴趣的是对所有输入停机的程序, 而非对某些输入停机的程序)。

易证明, 这两个问题都是 NP-完全的 (见习题 2.19)。注意到这两个 (判定) 问题并不互补 (即 (M, t') 可以同时是两个判定问题的回答为“是”的实例)。^②

有界不停机问题可能不可解 (即当 $P \neq NP$ 时不可解), 这一事实与程序验证的相关性要大于其与停机问题不可判定的相关性。原因是后者 (以及其他相关的不可判定问题) 涉及到任意长度的计算, 而前者只涉及明确限定了步数的计算。特别地, 有界不停机问题考虑的是是否存在一种输入, 使程序在某个给定时间界限内违反某种条件 (如停机)。

鉴于上述原因, 在一个计算机程序占主导地位的时代, 批评有界 (不) 停机是一种“不自然的”问题似乎显得非常奇怪 (然而, 我们在下一小节的标题中仍使用“自然的”这一传统术语, 它是指似乎与计算无关的自然计算问题)。

2.3.3 一些常见的 NP-完全问题

在肯定了 NP-完全问题的存在性之后, 我们要证明存在一些 NP-完全问题, 与问题定义中的计算并不 (明确地) 相关。我们强调人们已经发现了上千个这种问题 (参考文献[85]中列出了几百个 NP-完全问题)。

我们将证明判定命题公式的可满足性是 NP-完全的 (即 Cook 定理), 还指出某些组合问题也是 NP-完全的。目的是得到少部分 NP-完全性结论的样本, 以及开发证明新问题 NP-完全性的工具。

定理 2.19 证明中使用的归约被称为是“通用的”, 因为它 (显然) 适用于任意的 NP-问题。也就是说, 实际上, 提出了一种方法来构造从任意一个 NP-问题到某个已被证明的 NP-完全问题的归约, 而这样做完全符合 NP-完全性的定义。然而, 一旦我们知道了一些 NP-完全问题, 便开辟了另一条不同的道路。为了证明一个新问题的 NP-完全性, 可以将已知的 NP-

① 长度参数不一定要求等于时间上界。事实上, 该问题一个更常见的版本是给定两个上界 l 和 t , 并判定给定程序是否对于每个 l 比特输入都在 t 步内停机。易证明, 即使要求实例的两个参数之间满足关系 $t = p(l)$ 时, 这个问题仍是 NP-完全的 (这里 p 可以为任意多项式, 如 $p(n) = n^2$, 而不一定是正文中的 $p(n) = n$)。

② 事实上, (M, t') 不可能同时是这两个判定问题的“否”-实例, 但这也不表示两个问题互为补问题。事实上, 这两个判定问题将所有可能的实例 (M, t') 分为三种: 对任意输入 x (长度最多为 t), $M(x)$ 在 t 步内停机; 对某些输入停机; 对任何输入 M 都不会在 t 步内停机。注意: 第一种类型的实例恰好是有界不停机问题的“否”-实例, 而第三种实例恰为有界停机问题的“否”-实例。

完全问题归约到这个新问题。实际上，通常人们都采用这种思路，而它的成立是基于下面这个简单的命题。

命题 2.20 如果一个 NP-完全问题 Π 可以归约为 NP 中某个问题 Π' ，则 Π' 也是 NP-完全的。进一步地，在这里保留了 Karp-归约（或 Levin-归约）的可归约性。

证明：这个命题的证明可归结为归约的传递性。具体地，问题 Π 的 NP-困难性意味着 NP 中任意一个问题都可归约到 Π ，从而也可归约到 Π' 。因此，通过归约的传递性（见习题 2.6），NP 中任意问题都可归约到 Π' ，所以， Π' 是 NP-困难的。 ■

2.3.3.1 电路与表达式的可满足性：CSAT 与 SAT

考虑两个相关的计算问题：CSAT 和 SAT，其判定版本分别是指布尔电路和表达式的可满足性（建议读者复习一下 1.2.4.1 节中关于布尔电路、表达式和 CNF 范式的定义）。

教学注释：在证明 SAT 的 NP-完全性时，我们是在证明了电路可满足性（CSAT）的 NP-完全性之后，通过归约来证明的。这样可以分开处理 SAT 的 NP-完全性证明中最重要的两个部分：用电路来模仿图灵机，以及用带有辅助变量的公式来模仿电路。

CSAT

布尔电路是一种有向非循环图，其内部结点称为门（Gates），标记为某种（2 元或 1 元）布尔运算，外部结点称为顶点，与输入或输出相关联。为电路设置输入后，以自然方式为所有内部结点赋值，最后得到的输出就称为电路对给定输入的赋值。电路 C 对输入 z 的赋值记作 $C(z)$ 。我们感兴趣的是只有一个输出的电路，令 CSAT 表示所有可满足的布尔电路集合（即电路 C 属于 CSAT，如果存在一个输入 z ，使得 $C(z)=1$ ）。还需考虑相关的二元关系 $R_{\text{CSAT}} = \{(C, z) : C(z)=1\}$ 。

定理 2.21（CSAT 的 NP-完全性） 集合 CSAT（或关系 R_{CSAT} ）是 NP-完全的（PC-完全的）。

证明：易证 $\text{CSAT} \in \text{NP}$ （或 $R_{\text{CSAT}} \in \text{PC}$ ），因此这里只证明 NP-困难性。重点考虑判定版本（但是也讨论了搜索版本）。

我们再次构造一种通用归约（在本书中这是最后一次），将任意的 NP 问题归约到 CSAT。1.2.4.1 节中提出，多项式时间算法的计算过程可以用多项式规模电路来模仿，据此构造归约。在这里要模仿某个固定机器 M 对于输入 (x, y) 的计算，其中 x 为常量， y 为变量（但 $|y| = \text{poly}(|x|)$ ，且 $M(x, y)$ 计算的总步数为 $|x| + |y|$ 的多项式）。因此， x 将被“硬嵌入”到电路中，而将 y 作为电路的输入。电路自身，记作 C_x ，包含若干“层”，每一层代表着机器 M 的即时配置。在计算中相继配置间的关系（“非一致地”）由电路中的本地组件表示。电路的层数取决于限制 M 运行时间上界的多项式，另外还有一个附加组件用于检测最后一个配置是否被接受。因此，电路 C_x 中只有第一层（表示初始配置，其输入以 x 为前缀）依赖于 x 。关键在于判定对于给定 x ，是否存在一个 y （ $|y| = \text{poly}(|x|)$ ）使得 $M(x, y)=1$ （在给定步骤内）的问题可以归约为，是否存在 y ，使得 $C_x(y)=1$ 。在任意相应于关系 $R \in \text{PC}$ 的机器 M_R 上实现这种归约，就能证明 CSAT 的 NP-完全性。证明细节如下。

我们的目标是将任意集合 $S \in \text{NP}$ 归约到 CSAT。这里沿用定理 2.19 的证明中所定义的 R 、

p_R 、 M_R 和 t_R (即 R 为 S 的证据关系, p_R 是 NP-证据的长度上界, M_R 是判定 R 中成员的机器, t_R 为多项式时间界限)。不失一般性 (并且为简单起见), 假设 M_R 为单带图灵机。我们构造一个 Karp-归约, 将 S 的实例 x 映射到一个电路, 记作 $f(x) = C_x^{\text{def}}$, 使得 $C_x(y) = 1$ 当且仅当 M_R 在 $t_R(|x| + p_R(|x|))$ 步内接受输入 (x, y) , 从而有 $x \in S$ 当且仅当存在 $y \in \{0, 1\}^{p_R(|x|)}$ 使得 $C_x(y) = 1$ (即当且仅当 $C_x \in \text{CSAT}$)。电路 C_x 同时受 x 、 M_R 、 p_R 和 t_R 的影响 (其中 M_R 、 p_R 和 t_R 是固定的, 而 x 为变量)。

在描述电路 C_x 之前, 首先考虑 M_R 对于输入 (x, y) 的一种可能计算, 其中 x 固定, y 表示长度至多为 $p_R(|x|)$ 的字符串。计算将进行 $t = t_R(|x| + p_R(|x|))$ 步, 相应地, 产生 $t+1$ 个即时配置, 每个具有长度 t 。每种配置可以编码为 t 个符号对, 其中第一个符号表示当前存储的内容, 第二个符号或者表示机器的状态, 或者表示机器不在当前位置这一事实。因此, 每对符号属于 $\Sigma \times (Q \cup \{\perp\})$, 其中 Σ 为机器 M_R 的有限的“工作字母表”, Q 为内部状态集合, $\perp$ 为指示机器不在存储单元的符号。初始配置将 xy 作为输入, 判定结果 $M_R(x, y)$ 可以由最后一种配置 (最左边的存储单元) 中读出。^①除了第一行以外, 每行中的项都由上一行中的对应项来确定, 确定的依据是 M_R 的迁移函数。另外, 当前行中每一项的值由上一行中 (最多) 三项的值来确定 (见习题 2.20)。因此, 上述计算过程可以表示为一个 $(t+1) \times t$ 的阵列, 其中的每一项对常数可能性中的一个编码, 而这一项又可编码为固定长度的比特串, 如图 2.1 所示。

电路 C_x 的结构对应于上述的阵列。每一项用固定数量的门表示, 当 C_x 要计算 y 时, 为这些门分配相应的值, 这些值就是当前项中内容的编码。特别地, 阵列中第一行中的各项编码是归约输入 (即 x) 的“硬连接”, 并将电路的输入 (即 y), 赋值给足够多的输入端。也就是说, 电路有 $p_R(|x|)$ 个 (“真实”) 输入 (对应于 y), 而将常数硬连接到表示第一行的其他 $O(t - p_R(|x|))$ 个门, 这个过程可由简单的组件 (图 1.3) 实现。事实上, 第一行中额外的硬连接对应于初始配置中其他固定元素 (如空符号, 以及对初始状态和初始位置的编码, 如图 2.1 所示)。后续行中各项的“编码” (或在求值中的计算) 利用了常数规模的电路, 该电路基于上一行中相应的三项来确定当前项的值。回顾每一项被编码为常数个门, 因此, 这些常数规模的电路仅能计算习题 2.20 中描述的常数规模函数。另外, 电路 C_x 还有几个附加门, 用于检查最后一行中各项的值, 从而判定其是否为可接受配置的编码。^②注意到电路 C_x 可由串 x 出发, 在多项式时间内构造, 因为只需要用一种适当方式对 x 编码, 并生成一个“强一致的”, 规模为 $O(t_R(|x| + p_R(|x|))^2)$ 的类似于网格的电路即可。^③

虽然上述 C_x 的构造中利用了 (单带) 图灵机的各种具体细节, 但它易被改造成适用于其他高效的计算模型 (只需证明在这些模型中, 一种配置到后续配置的变换可用一个 (多项式时间构造的) 电路来模仿)。^④另外, 回顾 Cobham-Edmonds 定理, 其中称任意一个在多项式

① 这里使用 1.2.3.2 节中输入的表达式, 其中输入被写入最左边的单元, 而机器在输入的右边单元停机。

② 注意到只需检查最后一行最左边的两项。这里假设电路对于停机配置可以一直计算到最后一行。另外, 可以检查整个阵列中可接受/停机配置的存在性, 因为这个条件是相当简单的。

③ 已知有一种更高效的构造, 可以生成几乎线性规模的电路 (即电路规模为 $\tilde{O}(t_R(|x| + p_R(|x|)))$), 见参考文献[180]。

④ 事实上, 在所有自然的计算模型 (如 RAM 模型) 下构造这样的电路并不复杂, 其中每个比特在下一配置中可以表示为关于前面配置的简单布尔表达式。

的合取, 这些“浅”电路具有无限扇入, 并且可以表示为 CNF 范式。上述的辅助变量保留原始电路内部的可能取值, 而 CNF 中的子句可以实现这些值与相应的电路门运算间的一致性。例如, 如果门 $gate_i$ 和 $gate_j$ 组成门 $gate_k$, 而后者是一个合取门, 则相应的辅助变量 g_i 、 g_j 、 g_k 应该满足条件 $g_k \equiv (g_i \wedge g_j)$, 这可以写成一个含 4 个子句的 3CNF, 细节如下:

我们将 CSAT Karp-归约为 SAT。给定有 n 个输入端和 m 个门的布尔电路 C , 首先构造出 m 个常数规模的, 含 $n+m$ 个变量的表达式, 其中前 n 个变量对应于电路的输入端, 其他 m 个变量对应于电路中的门。第 i 个表达式取决于对应着第 i 个门的变量, 以及相应于输入到这些门的顶点的 1-2 变量 (即 2 顶点对应于 “ $\wedge$ ” 门和 “ $\vee$ ” 门, 而 1-顶点对应于 “ $\neg$ ” 门, 这些顶点或者为输入的端点, 或者为其他门)。这个 (常数规模的) 表达式可被一种真值赋值满足, 当且仅当该赋值与各个门的功能匹配 (即向门中输入相应值时能得到对应的输出)。注意到这里常数的表达式可以写成常数规模的 CNF 范式 (实际上是 3CNF 范式)。①对这 m 个表达式与连接输出端的门相关的变量求合取, 可以得到一个 CNF 表达式 Φ (见图 2.2, 其中 $n=3$, $m=4$)。

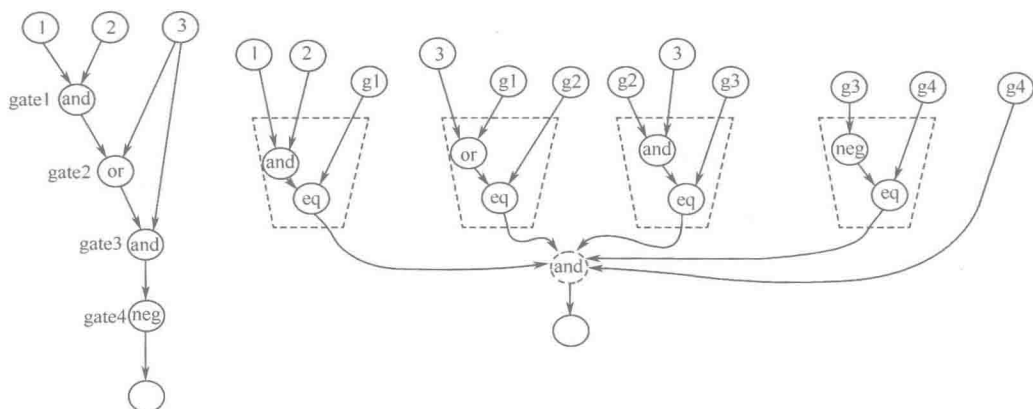


图 2.2 利用辅助变量 (即 g_i) 可以“分割”一个深度为 5 的电路 (分为 CNF 范式)。虚线内的区域将被替换为等价的 CNF 范式。虚线圈表示一个无界扇入、用与门表示常数规模电路合取的电路 (可实现初始门的功能), 且其中变量表示输入为原始电路输出端的门

注意到 Φ 可以在多项式时间内由电路 C 构造, 也就是说, C 到 $\Phi = f(C)$ 的映射是多项式时间可计算的。我们断言 C 属于 CSAT 当且仅当 Φ 属于 SAT。

(1) 假设存在某个串 s , 使得 $C(s)=1$, 则为第 i 个辅助变量分配电路 C 在计算 s 时第 i 个门的值, 可以得到 (与 s 一起) 满足 Φ 的一种真值赋值。这是由于这种赋值满足所有 m 个常数规模的 CNF 范式以及与 C 的输出相关联的变量。

(2) 另一方面, 如果 τ 满足 Φ , 则令 τ 中前 n 个比特对应于使电路 C 输出为 1 的输入。这是由于 m 个常数规模的 CNF 范式保证了为 Φ 中变量所分配的值对应于 C 对 τ 中前 n 比特的计算结果, 而与 C 的输出相关联的变量取值为“真”则保证了 C 的计算输出为 1。

注意: 后一个映射 (由 τ 到其 n 比特前缀) 就是 Levin-归约的定义中所要求的第二个映射。

① 回顾任意布尔函数可被写成 CNF 范式, 其规模为输入长度的指数, 在该例中为常数 (即 2 或 3)。事实上, 注意这里的布尔函数依赖于 2~3 个布尔变量 (因为它们指示着相应赋值是否遵循了函数功能)

至此,我们证明了 f 是 CSAT 到 SAT 的一个 Karp-归约,而为 f 增加上述的第二个映射之后,可以得到由 R_{CSAT} 到 R_{SAT} 的 Levin-归约。

说明

在 Karp-归约的正确性证明中,明确要求相应 Levin-归约定义中的第二个映射,这很正常。实际上(见习题 2.28),在 Karp-归约的典型叙述中,提供了两个辅助的多项式时间计算的映射(除了由一个问题(如 CSAT)的实例到另一个问题(如 SAT)的实例间的主映射以外):第一个辅助映射将原像问题(如 CSAT)实例的解映射为归约的像问题(如 SAT)的实例的解,而第二个映射作用恰好相反(注意:在 Levin-归约的定义中只要求有主映射和第二个辅助映射)。例如,在由 CSAT 到 SAT 的 Karp-归约(记作 f)的正确性证明中,规定了两个额外的映射 h 和 g ,使得 $(C,s) \in R_{\text{CSAT}}$ 蕴含了 $(f(C),h(C,s)) \in R_{\text{SAT}}$ 以及 $(f(C),\tau) \in R_{\text{SAT}}$ 蕴含了 $(C,g(C,\tau)) \in R_{\text{CSAT}}$ 。具体地,在定理 2.22 的证明中,使用 $h(C,s)=(s,a_1,\dots,a_m)$,其中对 a_i 的赋值是 $C(s)$ 在计算中第 i 个门的值,而 $g(C,\tau)$ 是 τ 的 n -比特前缀。

3SAT

注意:在定理 2.22 的证明中,由 Karp-归约所导出的表达式是合取范式(CNF),其中每个子句最多含 3 个变量。因此,上述的归约实际上证明了 3SAT 的 NP-完全性(即限制为 CNF 形式的 SAT,且每个子句最多含 3 个变量)。作为代替方案,可以将 SAT(即 CNF 范式的可满足性)Karp-归约到 3SAT(即 3CNF 范式的可满足性),方法是通过引入辅助变量,用含 3 变量的子句来代替长子句(见习题 2.21)。无论用何种方法,可以得到如下结论,而其中的附加部分可以用附加的归约来证明。

命题 2.23 3SAT 是 NP-完全的。另外,若将问题实例局限于每个变量至多出现在 3 个子句中的情形,该问题仍是 NP-完全的。

证明概要:后一部分可以通过将 3SAT 归约到该问题而证明。我们只需在变量每次出现时用一个新的变量副本代替原变量即可,并增加一些子句来保证为所有这些副本分配了相同的值。具体来讲,如果用副本 $z^{(1)}, z^{(2)}, \dots, z^{(m)}$ 代替变量 z ,则增加的子句为 $z^{(i+1)} \vee \neg z^{(i)}$, $i=1, 2, \dots, m$ (这里 $m+1$ 相当于 1)。

相关问题

注意到 SAT 的实例可以看作是施加于布尔变量之上的布尔条件组。这些条件可以用各种类型的算术条件组来模仿,这蕴含了解决后一种类型的问题是 NP-困难的。这方面的例子包括整系数线性方程组(见习题 2.23)及二次方程组(见习题 2.25)。

2.3.3.2 组合论及图论问题

教学注释:本小节介绍 NP-完全性理论中的一些结论和证明技巧(如在计算问题间构造归约的技巧)。作者相信这些传统教学内容极富见解,但在复杂性的课程中却常常被忽略。

组合论中有许多有趣的问题,我们只介绍其中几个被认为是 NP-完全的。这里只讨论各种问题的判定版本,并采用不太正式的风格。具体地,我们会将判定问题典型地描述为判定其给定实例(属于一类相关的实例集)为“是”实例还是“否”实例(而不是判定任意给定串是否属于“是”实例集)的问题。2.4.1 节中进一步讨论了这种叙述风格并对其进行了严格的形式化。我们还跳过了这些判定问题属于 NP 的证明,事实上,对于 NP 中自然的问题, NP 成员的证明是相当直接的。

集合覆盖 (Set Cover)

我们从集合覆盖问题开始,其实例包含一组有限集 $S_1, S_2, \dots, S_m$ 和整数 K , 问是否存在 (最多) K 个集合, 可以覆盖 $\bigcup_{i=1}^m S_i$ (即是否存在下标集 $i_1, i_2, \dots, i_K$ 使得 $\bigcup_{j=1}^K S_{i_j} = \bigcup_{i=1}^m S_i$)。

命题 2.24 集合覆盖问题是 NP-完全的。

证明概要: 构造一种从 SAT 到集合覆盖的归约。对于含 m 个子句和 n 个变量的合取范式 Φ , 考虑集合 $S_{1,t}, S_{1,f}, \dots, S_{n,t}, S_{n,f} \subseteq \{1, 2, \dots, m\}$, 其中 $S_{i,t}$ (或 $S_{i,f}$) 是将第 i 个变量设为真 (或假) 时, 可满足的子句的下标集合。也就是说, 如果第 i 个变量在第 j 个子句中未取反 (或取反), 则 $j \in S_{i,t}$ (或 $j \in S_{i,f}$)。注意到所有这 $2n$ 个集合的并集等于 $\{1, 2, \dots, m\}$ 。现在, 对于输入 Φ , 归约过程输出集合覆盖问题的实例 $f(\Phi) = ((S_1, S_2, \dots, S_{2n}), n)$, 其中 $S_{2i-1} = S_{i,t} \cup \{m+i\}$, 而对 $i = 1, 2, \dots, n$, $S_{2i} = S_{i,f} \cup \{m+i\}$ 。

注意: f 是多项式时间可计算的, 并且如果 Φ 对于 $\tau_1 \tau_2 \dots \tau_n$ 可满足, 则集合 $\{S_{2i-\tau_i} : i = 1, 2, \dots, n\}$ 能覆盖 $\{1, 2, \dots, m+n\}$ 。因此, $\Phi \in \text{SAT}$ 蕴含着 $f(\Phi)$ 为集合覆盖问题的“是”实例。另一方面, 对任意 i , 集合 $\{m+1, \dots, m+n\} \subset \{1, 2, \dots, m+n\}$ 的任意一种覆盖中必须包含 S_{2i-1} 或 S_{2i} 。因此, 包含了 n 个集合 S_j 的 $\{1, 2, \dots, m+n\}$ 的一种覆盖中必含 S_{2i-1} 或 S_{2i} , 但不会两者都有。相应地, 通过设置 τ_i 的值 (即 $\tau_i = 1$ 当且仅当 S_{2i-1} 属于覆盖), 可以暗示 $\{S_{2i-\tau_i} : i = 1, 2, \dots, n\}$ 覆盖了 $\{1, 2, \dots, m\}$, 这反过来又蕴含着 $\tau_1 \tau_2 \dots \tau_n$ 满足 Φ , 因此, 如果 $f(\Phi)$ 为集合覆盖问题的“是”实例, 则 $\Phi \in \text{SAT}$ 。□

恰当覆盖和 3XC

恰当覆盖与集合覆盖相似, 不同之处在于规定了覆盖中使用的子集不相交。即全集中每个元素都恰好由覆盖中的一个集合所覆盖。如果将实例集限制为子集中只含 3 个元素, 则得到了 3-恰当覆盖问题 (3XC), 这里没有必要规定覆盖中的集合个数。3XC 问题相当复杂, 但在证明其他问题的 NP-完全性中非常有用 (将 3XC 归约到这些问题)。

命题 2.25 3-恰当覆盖问题是 NP-完全的。

事实上, 由这个命题可直接得出恰当覆盖问题 (其中允许任意大小的集合) 是 NP-完全的。也可以得到覆盖中集合个数未规定以及规定了集合个数界限的各种情况 (如规定了上界或下界或两者都规定)。

证明概要: 我们利用 3 个归约的合成来构造归约。首先将 3SAT 的受限情况归约到集合覆盖问题的受限版本, 记作 3SC, 其中每个集合至多含 3 个元素 (而通常一个实例中包含有限集序列和整数 K)。具体地, 我们针对的是 3SAT 的这样一种实例, 其中每个变量至多出现在 3 个子句当中, 而前面提到这个受限的问题是 NP-完全的 (见命题 2.23)。实际上, 可以进一步地将 3SAT 的这种简化成每个变量至多出现在两个子句当中。^②现在, 利用命题 2.24 证明中的归约, 将 3SAT 的这种新版本归约到 3SC, 并注意到在受限实例中每个集合的势最多为 3 (即比每个变量出现的次数大 1)。

下面将 3SC 归约到以下恰当覆盖问题的受限情况, 记作 3XC', 其中每个集合最多含 3 个元素, 该问题的实例包含一组有限集和整数 K , 问是否存在一个最多含 K 个集合的恰当覆

① 显然, 集合覆盖的两种定义 (即要求恰好有 K 个集合, 或至多有 K 个集合) 在计算上是等价的。

② 观察到如果一个变量的 3 次出现均为同一种形式 (即都取非, 或不取非), 则为这个变量的赋值可以满足所有包含它的子句, 从而该变量及包含它的子句可以从实例中省略掉。这就将每个变量至多出现 3 次的 3SAT 的实例缩小为每个变量最多出现在两个子句中。实际上, 进一步观察命题 2.23 的证明, 会发现这个简化的实例也满足后一条性质。

盖。归约将 3SC 的实例 $((S_1, S_2, \dots, S_m), K)$ 映射为实例 (C', K) , 其中 C' 是 $S_1, S_2, \dots, S_m$ 中所有集合的子集的集合。由于每个 S_i 最多含 3 个元素, 其非空子集数最多为 7, 因此, 上述归约可以在多项式时间内计算。读者易验证归约的正确性。

最后, 将 $3XC'$ 归约到 $3XC$ 。考虑 $3XC'$ 的一个实例 $((S_1, S_2, \dots, S_m), K)$, 并假设 $\bigcup_{i=1}^m S_i = [n]$ 。如果 $n > 3K$, 则显然是一个“否”实例, 可将这种实例映射为 $3XC$ 的哑“否”实例, 从而可以假设 $x = 3K - n \stackrel{\text{def}}{\geq} 0$ 。注意: x 表示含 K 个集合, 每个集合含 3 个元素的恰当覆盖“超出”覆盖的能力。因此, 可在集合系统中增加 x 个新元素, 记作 $n+1, \dots, 3K$, 并将所有满足 $|S_i| < 3$ 的 S_i 用覆盖了 S_i 及 $\{n+1, \dots, 3K\}$ 中任意元素的 3 元素集合组代替。换言之, 当 $|S_i| = 2$ 时, 用集合 $(S_i \cup \{n+1\}, \dots, S_i \cup \{3K\})$ 来代替 S_i , 其中每个单独的 S_i 用集合 $S_i \cup \{j_1, j_2\}$ 代替, 这里 $j_1, j_2 \in \{n+1, \dots, 3K\}$ 且 $j_1 < j_2$ 。另外, 我们还加入 $\{n+1, \dots, 3K\}$ 的所有可能的 3 元素集合。这就完成了第 3 个归约, 其正确性验证留给读者完成。 $\square$

顶点覆盖, 独立集和团

现在讨论来自图论的问题 (见附录 G.1), 先介绍顶点覆盖, 它是集合覆盖问题的特例。其实例包含一对 (G, K) , 其中 $G = (V, E)$ 为简单图, K 为整数, 判断是否存在 (最多) 含 K 个顶点的集合, 与图中每条边都相关联 (即 G 中每条边至少有一个端点在该集合中)。事实上, 顶点覆盖问题的实例可以看作是集合覆盖问题的实例, 只需考虑集合 $(S_v)_{v \in V}$ 即可, 其中 S_v 表示与顶点 v 相关联的边集 (即 $S_v = \{e \in E : v \in e\}$)。因此, 集合覆盖的 NP-困难性可由顶点覆盖的 NP-困难性直接得出 (但是在这里似乎没有什么帮助: 因为我们已经知道集合覆盖是 NP-困难的, 而要证明的是顶点覆盖的 NP-困难性)。还要注意的, 顶点覆盖与独立集和团在计算上等价 (见习题 2.26), 因此只需证明三者之一的 NP-困难性即可。

命题 2.26 顶点覆盖、独立集和团是 NP-完全问题。

教学注释: 以下归约不是“标准”形式 (见习题 2.27)。它由 FGLSS-归约 (见习题 9.18) 变换而来, 在这里用于引出后者。另外, 虽然归约的目的是要生成一个大图, 但我们发现它比“标准”形式的归约更清晰。

证明概要: 构造由 3SAT 到独立集问题的归约。对于含 m 个子句 n 个变量的 3CNF 范式 Φ , 构造一个含 $7m$ 个顶点的图, 记作 G_Φ 。图中的顶点组成 m 个团, 对应于 Φ 中的 m 个子句, 每个团中包含 7 个顶点, 对应于 7 种满足该子句的真值赋值 (对于子句中的 3 个变量的赋值)。除了 m 个团中的边外, 在每对顶点间增加一条边, 对应于相互不一致的部分赋值, 即如果对第 i 个子句的某种特定的 (可满足的) 赋值与对第 j 个子句的某种 (可满足的) 赋值不一致, 则在相应顶点间增加一条边 (注意: m 个团的内部边可以看作是连接相互不一致的部分赋值的边的特殊情况)。因此, 对于输入 Φ , 归约输出一对 (G_Φ, m) 。

注意: 如果赋值 τ 满足 Φ , 则在由 τ 诱导出对应于部分可满足赋值的 m 个顶点间不存在边 (我们强调 Φ 的任意一种真值赋值都能产生一个独立集, 但是只有可满足的赋值才能保证这个独立集中包含了来自 m 个团的顶点)。因此, $\Phi \in \text{SAT}$ 蕴含着 G_Φ 中有一个规模为 m 的独立集。另一方面, G_Φ 中规模为 m 的独立集中, 其每个顶点恰好来自一个团, 因此诱导出 Φ 的一种可满足的赋值 (我们强调每个独立集可以诱导出 Φ 的一种一致的真值赋值, 因为在各种团中选择的部分赋值必须是一致的, 并且一个包含了来自特定团的顶点的独立集可以诱导出满足相应子句的赋值)。因此, 如果 G_Φ 中有规模为 m 的独立集, 则 $\Phi \in \text{SAT}$ 。 $\square$

图的三可着色性 (G3C)

问题的实例为一个图，问是否可用 3 种颜色对其着色，使得相邻顶点不会分配同一种颜色。

命题 2.27 图的 3 可着色问题是 NP-完全的。

证明概要：将 3SAT 映射到 G3C 问题，方法是将 3CNF 范式 Φ 映射为图 G_Φ ，其中包含两个特殊的（“指定”）顶点， Φ 中的所有变量， Φ 中的所有子句，以及连接这些成分的边。

- 两个指定顶点被称为 **ground** 和 **false**，由一条边相连，保证在 G_Φ 的任意一种 3 着色方案中它们的颜色不同。将顶点 **ground**（或 **false**）的颜色称为 **ground**（或 **false**），而第三种颜色称为 **true**。

- 与变量 x 有关的部分是一对顶点，与 x 的两种文字（即 x 和 $\neg x$ ）相关联。这些顶点之间有边相连，且其中任何一个都与顶点 **ground** 相连。因此，在 G_Φ 的一种 3-着色方案中，与该变量相关联的一个顶点被着色为 **true**，而另一个被着色为 **false**。

- 图 2.3 中描绘了与子句 C 相关联的部分。其中包含一个 **master** 顶点，记作 M ，以及 3 个终端顶点，记作 T_1 、 T_2 和 T_3 。**master** 顶点与 **ground** 和 **false** 都有边相连，因此，在 G_Φ 的 3-着色方案中，**master** 顶点必须被着色为 **true**。这一部分具有这样的性质，即有可能为终端顶点分配颜色 **true** 和 **false** 的任意组合，但不会为所有的终端分配颜色 **false**。与子句 C 相关联的终端可以由与 C 中相应变量相关联的顶点来识别。这意味着与各子句相关的部分中的顶点并非完全不相连，而是可能会共享某些终端（以及其自身的变量部分），^①如图 2.4 所示。

归约的正确性验证留作习题。

□

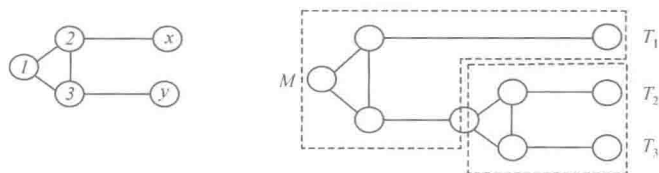


图 2.3 子句分图及其子图。在对子图的普通 3-着色方案中，如果 $x = y$ ，则必有 $x = y = 1$ 。

因此，如果分图中 3 个终端被分配同一种颜色 x ，则 M 也被分配颜色 x

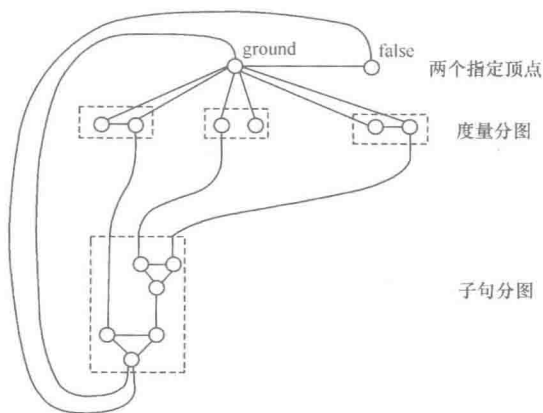


图 2.4 单个子句分图及相关变量

^① 相反，可以使用不相连的子图，并将每个终端与相应的文字（在相应的顶点部分）“连接起来”。这种连接（即一个辅助的子图）可以强制性地令两个端点在任意一种 3-着色方案中颜色相同。

2.3.4 既不属于 P 也非 NP-完全的 NP 集

如 2.3.3 节所述, 人们已经证明了上千个 NP-完全问题 (见参考文献[81]中的附录, 其中列出了 300 多个主要的 NP-完全问题), 从而期望任意的 NP 集或者是 NP-完全的, 或者属于 P 。然而, 这种天真的想法是错误的: 假设 $NP \neq P$, 则 NP 中存在一些集合, 既非 NP-完全, 也不属于 P , 这里的 NP-困难性可以通过 Cook-归约来证明。

定理 2.28 假设 $NP \neq P$, 则在 $NP \setminus P$ 中存在集合 T 使得 NP 中某些集合不能 Cook-归约为 T 。

定理 2.28 指出, 如果 $NP \neq P$, 则 NP 类可分为 3 个非空子类: P 类; 可由 NP 类经 Cook-归约得到的问题类以及剩余的问题类, 记作 NPI 。已知前两类是非空的, 定理 2.28 则在 $NP \neq P$ 的条件下, 确认了 NPI 也是非空的。这实际上是一个必要条件 (因为当 $NP = P$ 时, NP 中任意一个集合都可以 Cook-归约为 NP 中其他集合)。

以下对定理 2.28 的证明中构造了 NPI 中一个非自然的判定问题。这里要指出人们猜测某些自然的判定问题 (如在计算上与分解整数等价的问题) 属于 NPI 。另外, 如果 $NP \neq coNP$, 其中 $coNP = \{ \{0,1\}^* \setminus S : S \in NP \}$, 则有 $\Delta \stackrel{\text{def}}{=} NP \cap coNP \subseteq P \cup NPI$ 成立 (作为定理 2.35 的推论)。因此, 如果 $NP \neq coNP$, 则 $\Delta \setminus P$ 为 NPI 的 (自然) 子集, 而当 $\Delta \neq P$ 时 NPI 非空。回顾定理 2.28 似乎在更弱的假设即 $NP \neq P$ 下表明 NPI 是非空的。

教学注释: 建议不加证明地讲述定理 2.28 或仅提供证明思路。

证明概要: 基本思想是对 $NP \setminus P$ 中任一集合进行修改, 使所有可能的归约 (从 NP 到修改后的集合) 以及所有可能的多项式时间判定程序 (对于修改后的集合) 均失效。具体地, 从集合 $S \in NP \setminus P$ 出发, 得到集合 $S' \subset S$, 使得一方面不存在从 S 到 S' 的多项式时间归约, 另一方面 $S' \in NP \setminus P$ 。将 S 修改为 S' 的过程需要迭代, 以使可能的归约失效 (通过从 S 剩余的部分中去掉足够多的串), 并使可能的判定程序失效 (通过添加 S 剩余的部分中足够多的串)。具体地, 从 S 到 S' 的每个可能的归约都可通过去除当前 S' 中的有限个元素而失效, 而每个可能的判定程序可通过保留当前 S' 中有限个元素而失效。这两个断言建立于以下两个事实之上。

(1) (从不属于 P 的集合) 到任意有限集 (如 S 的一个有限子集) 的任意一种多项式时间归约一定会失效, 因为只有 P 中的集合才可以 Cook-归约到有限集。因此, 对任意有限集 F_1 以及任意可能的归约 (即多项式时间预言机), 存在一个输入 x , 使得得到 F_1 的归约失效。

我们强调当做出正面回答的询问均来自 F_1 时, 上述归约失效。另外, 这种失效是由于询问构成的有限集 (即对小于或等于 x 的输入调用归约时, 发出的所有询问构成的集合) 造成的。因此, 对任意有限集 $F_1 \subset S' \subseteq S$, 在去掉 S' 中有限个元素但不减少 F_1 中的元素时, 从 S 到 S' 的任意归约都会失效。

(2) 对任意有限集 F_2 , $S \setminus F_2$ 的任意多项式时间判定程序一定会失效, 因为 S 可以 Cook-归约到 $S \setminus F_2$ 。因此, 对任意可能的判定程序 (即多项式时间算法), 存在一个输入 x 使程序失效。

我们强调这种失效的原因是 $S \setminus F_2$ 的有限 “前缀” (即集合 $\{z \in S \setminus F_2 : z \leq x\}$)。因此, 对任意有限集 F_2 , 通过保留 $S \setminus F_2$ 的一个有限子集, 可使 $S \setminus F_2$ 的任意一个多项式时间判定程序

失效。

如上所述, 将 S 修改成为 S' 可通过迭代实现, 交替地使可能的归约失效 (通过在 S 的剩余部分中去掉有限个串), 或者使可能的判定程序失效 (通过向 S 的剩余部分中加入有限个串)。这个过程可以高效完成, 因为最基本的步骤是确定交替中第一个可能的点 (其中去掉 (或加入) 足够多的串以使下一个可能的归约 (或判定程序) 失效)。这足以保证交替过程中能高效地确定足够多的点 (虽然绝非最优点)。因此, S' 是 S 与 $\mathcal{P}$ 中某个集合的交集, 这蕴含着 $S' \in \mathcal{NP}$ 。关于上述思想的实现, 以下给出一些注解。

第一个问题是上述设计需要 (“高效地”) 列举出所有多项式时间预言机 (或多项式时间算法)。然而, 这些集合中任意一个都无法由 (一个算法) 列举。因此, 作为代替, 我们将模仿与所有可能的多项式相关的机器, 并对每个对 (M, p) , 考虑机器 M 的执行, 其时间上界由多项式 p 规定。也就是说, 由对 (M, p) 出发, 在 M 对输入 x 的执行中, 将 $p(|x|)$ 步之后的执行均挂起, 从而获得机器 M_p 。我们强调并不知道机器 M 是否运行于多项式时间, 但是任意多项式时间机器的计算都可被某个对 (M, p) “覆盖”。

下一步, 搞清楚在归约和判定程序中需要排除的过程。我们构造 $\mathcal{P}$ 中的 “过滤器” 集 F , 使得最终的集合 S' 等于 $S \cap F$ 。回顾我们需要选择 F 使得每个从 S 到 $S \cap F$ 的多项式时间归约失效, 且使判定 $S \cap F$ 的每个多项式时间算法失效。关键是对任意有限的 F' , 要使从 S 到 $S \cap F'$ 的每个多项式时间归约失效, 而对 F 的任意有限补集 (即有限的 $\{0, 1\}^* \setminus F$), 要使判定 $S \cap F'$ 的每个多项式时间算法失效。进一步地, 这些失效发生在某些输入中, 而这些输入可由 S 与 F 中的有限部分来确定。因此, 我们交替地使可能的归约或判定程序失效, 不会尝试去确定 “最优” 的变化点, 而是高效确定变化点 (这又需考虑 F 中成员的高效判定)。具体地, 令 $F = \{x: f(|x|) \equiv 1 \pmod{2}\}$, 其中 $f: \mathbb{N} \rightarrow \{0\} \cup \mathbb{N}$, 满足: 前 $f(n)-1$ 个机器中, 每个都对一些长度至多为 n 的输入失效; $f(n)$ 的值可在时间 $\text{poly}(n)$ 内计算。

$f(n)$ 的值可由以下过程来定义, 其中恰好包含 n^3 步计算 (这里的立方时间纯属偶然选择), 且以迭代方式进行 (迭代的次数预先未知), 在第 $i+1$ 次迭代中, 我们希望找到一个输入, 使得第 $i+1$ 个 (修改后的) 机器失效。具体地, 在第 $i+1$ 次迭代中, 以词典序扫描所有输入, 直到找到一个输入, 使第 $i+1$ 个 (修改后的) 机器失效, 当 $i+1$ 为奇数时, 该机器是一个预言机, 否则就是一个标准机器。如果检查到第 $i+1$ 个机器失效, 则将 i 的值加 1, 转向下一次迭代。到达了允许的计算步数 (即 n^3) 时, 停止, 并输出当前的 i (即当前的 i 作为 $f(n)$ 的值输出)。当然, 这个过程在很大程度上依赖于确定第 $i+1$ 个机器是否会对某些特定输入失效。直观上, 这些输入的长度远小于 n , 所以在时间 (大约为) n^3 内进行这些判定并非不可能——见下一段。

为了判定 (第 $i+1$ 个) 机器是否对某个特定输入 x 失效, 我们需要模拟该机器对 x 的计算过程, 并确定 x 是否属于相关的集合 (或者为 S , 或者为 $S' = S \cap F$)。回顾如果 $i+1$ 为偶数, 则需要使一个 (试图判定 S' 的) 标准机器失效, 而当 $i+1$ 为奇数时, 就需要使一个 (将 S 归约为 S' 的) 预言机失效。因此, 对于偶数 $i+1$, 我们需要判定 x 是否属于 $S' = S \cap F$, 而对奇数 $i+1$, 需判定 x 是否属于 S 以及是否某些其他的串 (以询问的方式出现) 属于 S' 。判定 $S \in \mathcal{NP}$ 的成员可在指数时间内完成 (通过穷举搜索并尝试所有可能的 NP-证据)。事实上, 这意味着在计算 $f(n)$ 时, 只需要完成对具有 (n 的) 对数长度的输入的处理, 而无论如何, 在 n^3 步内

(在词典序搜索中), 无法到达长度大于 $3\log_2 n$ 的串。至于 F 的成员判定, 要求对足够多的整数计算 f 。也就是说, 由于 n' 将远远小于 n (因为 $n' \leq \text{poly}(|x|) \leq \text{poly}(\log n)$), 可能需要对各种整数 n' 计算 $f(n')$ 的值。因此, $f(n')$ 的值只能递归地计算 (将递归的步骤计入总的运算步骤中)。^① 一个要点是, 在考虑输入 x 时, 我们可能只需要对 $\{1, 2, \dots, p_{i+1}(|x|)\}$ 求 f 的值, 其中 p_{i+1} 是第 $i+1$ 个 (修改后的) 机器运行时间的多项式上限, 且得到这个值需要至多 $p_{i+1}(|x|)^3$ 步运算。最后总结, 在判定输入 x 是否会 (使第 $i+1$ 个机器) 发生失效的过程中, 需要的运算步数上限是 $|x|$ 的一个 (指数) 函数。

如上所示, 这个过程在检查长度超过 $3\log_2 n$ 的输入之前, 已经执行了 n^3 步, 但是这并不重要。重要的是, f 是一个无界函数 (见习题 2.34)。另外, 由构造, $f(n)$ 可在 $\text{poly}(n)$ 时间内计算。□

说明: 定理 2.28 的证明实际上证实了, 对任意 $S \notin P$, 存在 $S' \notin P$ 使得 S' 可以 Karp-归约为 S , 但是 S 不能 Cook-归约为 S' 。^② 因此, 如果 $P \neq NP$, 则在 $NP \setminus P$ 中存在一个无限的集合序列 $S_1, S_2, \dots$, 使得 S_{i+1} 可以 Karp-归约为 S_i , 但 S_i 不能 Cook-归约为 S_{i+1} 。即对 NP 中的一切问题, 存在对问题的无限分级 (虽然是不自然的分级), 使得每个问题比前一个问题更“容易” (因为它可归约到前面所有的问题, 而反过来则不成立)。

2.3.5 对完全问题的思考

也许只有那些已经思考过本书思想, 或类似思想的人, 才有可能理解本书内容。因此, 本书并不是一本教科书。其目的在于使那些能理解的人获得乐趣。

路德维希·维特根斯坦, 逻辑哲学导论

本小节请读者和我们一起思考诸如“究竟是什么因素导致完全问题的存在”这种类型的问题。相应地, 本小节的写作风格也是比较简单和不严密的; 在整体上可以看作是一个没有标准答案的练习, 让读者思考意义模糊不清的文字背后的实质内容。

已知 NP-完全问题存在, 问题在于在我们定义的模型中, 是何种因素导致了完全问题的存在。当然, 首先应该记住, 完全性针对的是一类问题; 完全问题能对这类问题中的每一个进行“编码”, 且自身应该属于该类。由于第一个因素, 即作为一类问题的编码, 已经足以令人惊奇 (至少对外行是如此), 所以我们先对此做一番解释。我们确定相对于问题模型的两种基本模式, 在对任意 (无限) 问题类编码时, 这是至关重要的。

(1) 每个问题都涉及到由可能实例构成的无限集合。

(2) 每个问题的定义都使用有限的描述 (如, 对任意给定实例, 能枚举所有可能解的算法)。^③

① 对这个过程, 我们不提出效率更高的实现。也就是说, 在每次需要 $f(n')$ 时, 可以重新计算这个值 (而不是将其保存下来以备后用)。

② 所谓 (S' 到 S) 的 Karp-归约是指, 当 $x \in F$ 时, 将 x 映射为其自身, 否则, 将 x 映射为 S 的一个固定的“否”实例。

③ 这似乎是描述一个问题最自然的方法。另一种描述涉及到一个算法, 能识别所有正确的由实例和解构成的对 (如 NP 的定义)。然而, 在这里, 我们也考虑“非高效的”描述 (如对停机问题的描述)。

这两个模式似乎有点自相矛盾,然而,将其放在一起便构成了通用问题的定义,即一个具有形如 (D, x) 的实例的问题,其中 D 是对某个问题的描述,而 x 为该问题的实例(要对 x 求解)。直观上,这种通用问题可以对任意其他问题编码(条件是问题的模型与上述范例一致):如果能对该通用问题求解,则也能解任意其他问题。^①

注意到上述通用问题实际上关于所有问题类都是完全的,但是关于任意一个只包含(利用算法)可解问题的类则不是完全的(因为该通用问题不可解)。现在把注意力转到可解的问题类,我们要寻找通用问题的一个版本,使得关于这些类也是完全的。这里遇到的典型困难是,给定一个描述 D (作为通用问题实例的一部分),我们无法识别 D 是否是某个类 C 中问题的描述(因为描述问题不可解)。这一事实是相对的,^②因为如果通用问题要对 C 之外的问题实例求解,则直观上,它自身也不会属于 C 。

在解决上述困难之前,要注意上面的模型方法在广义的计算理论中十分重要。特别是,到此为止,我们尚未参考任何复杂性因素。然而,复杂性因素是解决上述困难的关键:主要思想是将任意一个描述 D 修改为 D' ,使得 D' 总是属于 C ,而当 D 属于 C 时, D' 与 D 一致(即此时它们恰好描述了同一个问题)。我们强调当 D 不属于 C 时, D' 可以是任意的描述(只要仍在 C 中)。在许多复杂性类中,这种修改是可以做到的。考虑两种不同的问题类,这两个类都用与问题有关的算法(可识别正确解的算法,如NP的定义)的时间复杂性来定义。

(1) 用一个时间受限的函数 t (如 $t(n)=n^3$)定义的类。此时,将任意算法 D 修改为算法 D' ,使得对输入的 x , D' 可以模拟 $D(x)$ 执行中的(至多) $t(|x|)$ 步。在通用问题修改后的版本中,实例由 (D, x) 变为 (D', x) 。这个版本可对上述问题类 C 中的任意问题编码。

但是,这个问题(通用问题版本)自身在 C 中吗?答案取决于相应计算模型的模拟效率以及 t 的增长速度。例如,对于三重指数(Triple-exponential)的 t ,答案一定为“是”,因为 $t(|x|)$ 个步骤可在时间 $\text{poly}(t(|x|))$ 内模拟(在任意合理的计算模型下),且 $t(|(D, x)|) > t(|x|+1) > \text{poly}(t(x))$ 。另一方面,在大多数合理的模型下,对 $t(|x|)$ 步的模拟需要时间 $\omega(t(|x|))$,且对任意多项式 t ,有 $t(n+O(1)) < 2t(n)$ 。

(2) 用一族无穷多个具有不同增长率的函数(如多项式)定义的类。当然,我们可以选择一个函数 t ,比上述函数簇中任意函数都增长得更快,但是此时得到的通用问题显然不会属于该类。

注意到,在这种情况下,完全问题事实上是十分惊人的,具体说来,它将与一个函数 t_0 相关联,而 t_0 比该类中某些函数增长得慢(如某个给定的多项式比其他多项式增长缓慢)。类似地,这意味着用于描述通用机器的算法应该比描述该类中其他问题的算法快。这种结论假设两个问题的实例长度(近似)相同,从而我们通过人为地增加问题实例描述的长度来有意地违背这种假设。例如,如果 D 与时间上限 t_D 相关,则通用问题的实例 (D, x) 可以表示为

① 然而,注意到通用问题是无法(用算法)求解的。因此,暗示的两个子句均为假。事实上,问题的概念在这一阶段相当含糊,它必定超出了所有可解问题的集合。

② 这里忽略了使用承诺问题的可能性,承诺问题确实可以避免这样的实例,且不要求任何人识别它们。事实上,使用承诺问题便可解决这个困难,但是下一段之后的讨论仍是有效的。

$\left(D, x, 1^{t_0^{-1}(D(|x|)^2)}\right)$, 对于 NP 类的情况使用 $t_0(n) = n$ 。

我们认为最后一项解释了 NP-完全问题的存在性。但是 SAT 的 NP-完全性又如何解释呢？

首先要注意 CSAT 的 NP-困难性可以由布尔电路能模拟算法这一事实直接得出。^①这一基本事实源于算法的定义（其中假设单步计算是简单的），且在任意合理的计算模型下都成立。因此，对任意的 D 和 x ，求串 y ，使得 $D(x, y) = 1$ 的问题可以被“编码”为求一个串 y ，使 $C_{D,x}(y) = 1$ ，其中 $C_{D,x}$ 为一个布尔电路，且易于从 (D, x) 中得到。与 CSAT 的 NP-困难性所体现的基本事实相反，SAT 的 NP-困难性取决于一个绝妙的技巧，其中将 CSAT 的实例编码为 SAT 的实例。

如上所述，证明 SAT 的 NP-困难性时，可将 CSAT 的实例编码为 SAT 的实例。类似地，其他一些新问题的 NP-困难性可通过对已知 NP-完全问题的实例进行编码而证明。典型地，这些编码在本地进行，将原始实例的一小部分映射为生成实例的子集。事实上，这些依赖于问题的实例子集是编码的核心。有了这些子集后，易验证其效果。因此，这些子集中的大多数都不足为奇，但是有上千个自然问题中都存在这种子集绝对是一个奇迹。

2.4 三个前沿性问题

本节讨论 3 个较前沿的话题。第一个是承诺问题，前面的章节中没有涉及这个问题（见 2.4.1 节）。第二个问题是优化的 NP 类搜索问题（见 2.4.2 节），第三个是 NP 类中集合的补集（CoNP）所构成的类。

教学注释：复杂性理论的基础教程中并不涉及这些问题，然而，如果时间充足，我们建议至少是从宏观的角度讨论一下它们。

2.4.1 承诺问题

承诺问题是搜索问题和判定问题的一种自然的推广，它明确考虑合法的实例（而不是将任意字符串当成合法实例）。如前所述，这就为自然的计算问题提供了一种更充分的定义（而实际上这种定义可以用于所有非正式的讨论中）。例如，在 2.3.3.2 节中，我们利用了“给定一个图和一个整数……”（或“给定一个集合组……”）的说法来提出这种问题。换言之，假设输入的实例具有某种形式（或更确切地，“向解题者承诺”实例具有某种形式）。事实上，我们声称在这些情况下，可以去掉假设而不影响问题的复杂性，但是避免对这种说法的形式化定义，细节如下。

教学注释：承诺问题的概念最早以判定问题的形式给出，并仅在判定问题中得到典型应用。然而，我们认为，搜索问题中也同样自然地存在承诺问题。

2.4.1.1 定义

在搜索问题环境下，承诺问题是一种要求有所放宽的形式，它只要求对一个预先确定的

^① CSAT 之所以属于 NP，源于可在多项式时间内对电路求值。

集合中的实例求解，该集合称为承诺（Promise）。对于解的高效验证也有类似要求。

定义 2.29（带承诺的搜索问题） 带承诺的搜索问题由一个二元关系 $R \subseteq \{0,1\}^* \times \{0,1\}^*$ 和一个承诺集 P 组成，这样的问题又称为带承诺 P 的搜索问题 R ，它满足以下条件。

- 称带承诺 P 的搜索问题 R 可以由算法 A 求解，如果对任意 $x \in P$ ，当 $x \in S_R = \{x: R(x) \neq \emptyset\}$ 时， $(x, A(x)) \in R$ ，否则 $A(x) = \perp$ ，这里 $R(x) = \{y: (x, y) \in R\}$ 。

A 对来自 P 中的输入计算时，其时间复杂性定义为 $T_{A|P}(n) = \max_{x \in P \cap \{0,1\}^n} \{t_A(x)\}$ ，其中 $t_A(x)$ 是 A 的运行时间，当 $P \cap \{0,1\}^n = \emptyset$ 时， $T_{A|P}(n) = 0$ 。

- 如果存在多项式时间算法能解带承诺 P 的搜索问题 R ，则称该问题属于 $\mathcal{PF}$ 类的承诺问题扩展。^①

- 如果存在多项式 T 及算法 A ，使得对任意 $x \in P$ 和 $y \in \{0,1\}^*$ ，算法 A 至少运行 $T(|x|)$ 步，并有 $A(x, y) = 1$ 当且仅当 $(x, y) \in R$ ，则称带承诺 P 的搜索问题 R 属于 $\mathcal{PC}$ 类的承诺问题扩展。

我们强调如果输入不符合承诺（即 $x \notin P$ ），则对解题者不做任何要求，特别地，此时算法会停机并输出错误（事实上，在搜索问题的标准形式化定义中，考虑的是承诺的平凡情况即 $P = \{0,1\}^*$ ）。^②除了上述动机，还要特别指出一类带承诺的搜索问题。这些搜索问题中，承诺是指某实例有解（如在上述定义中令 $P = S_R$ ，其中 $S_R = \{x: \exists y \text{ 使得 } (x, y) \in R\}$ ）。我们把这类搜索问题称为公正（Candid）搜索问题。

定义 2.30（公正搜索问题） 称算法 A 解相应于二元关系 R 的公正搜索问题，如果对任意 $x \in S_R$ （即对任意 $(x, y) \in R$ ），有 $(x, A(x)) \in R$ 。算法 A 的时间复杂度定义为 $T_{A|S_R}(n) = \max_{x \in P \cap \{0,1\}^n} \{t_A(x)\}$ ，其中 $t_A(x)$ 是 $A(x)$ 的运行时间，当 $P \cap \{0,1\}^n = \emptyset$ 时， $T_{A|S_R}(n) = 0$ 。

注意：当 $x \notin S_R$ 时不做任何要求，具体地，此时，算法 A 或者输出一个错误解（虽然解并不存在）或者运行时间多于 $T_{A|S_R}(|x|)$ 步。显然，当 $R \in \mathcal{PC}$ 时，第一种情况不会发生。另外，当 $R \in \mathcal{PC}$ 时，如果“已知”算法 A 的时间复杂性（如可以在 $\text{poly}(n)$ 时间内计算 $T_{A|S_R}(n)$ ），则可以将 A 修改为解（一般）搜索问题 R 的算法 A' （即停机并输出每个输入的正确解），其运行时间为 $T_{A'}(n) = T_{A|S_R}(n) + \text{poly}(n)$ 。然而，我们不一定会知道当前算法的运行时间。另外，在 2.4.2 节中将看到，总是已知算法运行时间的假设是天真的，也是不合理的。

带承诺的判定问题

在判定问题环境下，承诺是指这样一种放宽形式，即只要求对属于某个预先定义集合中的实例做出判断，该集合称为承诺。对高效验证的要求以类似方式给出。为了更标准地使用术语，我们所说的承诺问题是指带承诺的判定问题。更正式地讲，承诺问题将所有的串分为

① 此时，在定义 A 的时间复杂度时，是参照来自于 P 中的输入还是参照所有可能的串并不重要。假设 A 的（多项式）时间复杂度 T 来自于 P 中的输入定义，则我们可以修改 A ，使其对任意输入，最多运行 $T(|x|)$ 步之后就停机。这种改动只影响到 A 对不属于 P 的输入（这种输入也是合法的）的输出。这种修改可以在多项式时间内完成，只需计算 $t = T(|x|)$ 并模拟 $A(x)$ 在 t 步内的执行即可。类似的说明也可应用于 $\mathcal{PC}$ 、 $\mathcal{P}$ 和 $\mathcal{NP}$ 的定义。

② 这里我们是指 2.1.4 节中的形式化定义。

三类：“是”实例、“否”实例和违背承诺的实例。标准判定问题也是承诺问题的一种特例，此时所有的输入都是合法的（即承诺为平凡集）。

定义 2.31 (承诺问题) 承诺问题由一对不相交的串集合组成，记作 $(S_{\text{yes}}, S_{\text{no}})$ ， $S_{\text{yes}} \cup S_{\text{no}}$ 称为承诺。

- 称算法 A 解承诺问题 $(S_{\text{yes}}, S_{\text{no}})$ ，如果对任意 $x \in S_{\text{yes}}$ ，有 $A(x)=1$ ，而对任意 $x \in S_{\text{no}}$ ，有 $A(x)=0$ 。如果存在多项式时间算法能解某个承诺问题，则称其属于 $\mathcal{P}$ 类的承诺问题扩展。

- 承诺问题 $(S_{\text{yes}}, S_{\text{no}})$ 属于 $\mathcal{NP}$ 类的承诺问题扩展，如果存在多项式 p 及多项式时间算法 V ，使得以下两个条件成立。

(1) 完备性。对任意 $x \in S_{\text{yes}}$ ，存在长度不超过 $p(|x|)$ 的 y ，使得 $V(x, y)=1$ 。

(2) 合理性。对任意 $x \in S_{\text{no}}$ 及任意 y ，有 $V(x, y)=0$ 。

我们强调，对于多项式时间复杂度的算法而言，在定义其时间复杂度时是只根据满足承诺的输入还是所有的输入并不重要。因此， $\mathcal{P}$ 和 $\mathcal{NP}$ 的扩展类（如 $\mathcal{PF}$ 和 $\mathcal{PC}$ ）在这里也是不变的。

承诺问题间的归约

Cook-归约可以很自然地扩展到承诺问题，如果假定对于违背（作为归约目标的问题中的）承诺的询问将得到任意回答的话。^①也就是说，无论违背承诺的询问是否能得到应答，预言机都应该能解原始问题。这个要求与归约和承诺问题在概念上是一致的。前面提到归约可以理解为将任意解归约目标问题的程序当作子程序来调用的过程。但是，在承诺问题中，这样的算法可能会对违背承诺的实例做出任意行为。我们强调归约的主要性质在这里仍保留（见习题 2.35）：如果承诺问题 Π 可以 Cook-归约为一个多项式时间可解的承诺问题，则 Π 也可在多项式时间内求解。

要注意，一个复杂性类到承诺问题的扩展并不一定继承了标准类的“结构”特性。例如，与定理 2.35 相反， $\mathcal{NP} \cap \text{co}\mathcal{NP}$ 中存在一些承诺问题，使得 $\mathcal{NP}$ 中任意集合可以 Cook-归约为这些问题，见习题 2.36。当然，由习题 2.36 显然得不到 $\mathcal{NP}=\text{co}\mathcal{NP}$ ，见 2.4.1.2 节结尾的进一步讨论。

以下的讨论既针对承诺问题的判定版本也针对搜索版本。承诺问题提供了对于计算问题最直接的定义方式（如对关于图的计算问题，承诺要求输入是一个图）。除了承诺问题的上述应用，我们指出，它们也可用于定义将计算问题限制到实例子集的自然情形。具体地，这种限制的含义是：受限问题的承诺集是未受限问题承诺集的子集。

定义 2.32 (对计算问题的限定)

- 对任意 $P' \subseteq P$ 及二元关系 R ，称带承诺 P' 的搜索问题 R 是对带承诺 P 的搜索问题 R 的限定。

- 称承诺问题 $(S'_{\text{yes}}, S'_{\text{no}})$ 是承诺问题 $(S_{\text{yes}}, S_{\text{no}})$ 的限定，如果 $S'_{\text{yes}} \subseteq S_{\text{yes}}$ 且 $S'_{\text{no}} \subseteq S_{\text{no}}$ 。

例如，我们称 3SAT 是 SAT 的限定，针对的是将允许的实例限定为 3CNF 范式（而非任意的 CNF 范式）这一事实。这里的两个计算问题都是判定可满足性（或求一种可满足的赋值），

① 从而承诺问题间的 Karp-归约不允许发出违背承诺的询问。具体地，称承诺问题 $\Pi = (S_{\text{yes}}, S_{\text{no}})$ 可以 Karp-归约为承诺问题

$\Pi' = (S'_{\text{yes}}, S'_{\text{no}})$ ，如果存在多项式时间映射 f ，使得对任意 $x \in S_{\text{yes}}$ （或 $x \in S_{\text{no}}$ ）有 $f(x) \in S'_{\text{yes}}$ （或 $f(x) \in S'_{\text{no}}$ ）。

然而, 在 3SAT 中对实例集 (即承诺集) 有进一步的限制。一个受限问题永远不会比原始问题更难, 可以用受限问题能归约为原始问题 (通过恒等映射) 的事实来理解这一现象。

其他应用及某些限制

承诺问题除了上述两种普通应用外, 还为各种特定的计算复杂性概念及结论提供了充分定义。这方面的例子包括“唯一解”(见 6.2.3 节) 的概念, 以及用于解决各种近似任务的“缝隙问题”的定义 (见 10.1 节)。在所有这些情况下, 都可以用承诺问题讨论自然的计算问题, 并说明其固有复杂度。从而, 承诺问题 (以及这种问题构成的类) 的复杂度关系到自然的问题。因此, 证明属于某个类的承诺问题的不可解性 (如称 $\mathcal{NP}$ 中某个承诺问题不能被多项式时间算法求解) 意味着要证明相应类中标准问题的不可解性。相反, 如 2.4.1.1 节的结尾所暗示的, 承诺问题的结构特性可能在相应的标准问题类中并不成立 (如见习题 2.36)。事实上, 我们在这里的确对固有 (或绝对) 性质 (如不可解性), 与结构上 (或相对) 性质 (如可归约性) 进行了区分。

2.4.1.3 避免承诺问题的标准习惯

虽然承诺问题为许多自然的计算问题的表述提供了一个良好框架, 但我们在以前的章节中却尽量避免提到这个框架。这是由于前面考虑的所有计算问题都易于判定一个给定实例是否满足承诺。例如, 给定一个公式, 易判定其是否属于 CNF (或 3CNF)。实际上, 在讨论布尔表达式时, 这个问题已经出现了: 我们实际上得到的是一个串, 并假设它对表达式编码 (在某种预先确定的编码方案下), 所以这里的承诺 (对于自然的编码方案是易判定的) 是指输入串是某个表达式的有效编码。在任何情况下, 如果承诺可以被高效识别 (即可以在多项式时间内判定承诺成员), 则可以通过使用以下两种“别扭”的习惯来避免承诺。

(1) 将实例集扩展到所有的串 (并允许相应的“哑”实例存在平凡解)。例如, 对于搜索问题, 可以定义所有违背承诺的实例无解, 或规定其有平凡解 (如是其自身的解); 就是说, 对于带承诺 P 的搜索问题 R , 我们可以考虑 (标准) 搜索问题 R , 其中 R 被修改为: 对任意 $x \notin P$, $R(x) = \emptyset$ (或对任意 $x \notin P$, $R(x) = \Phi$)。在 (判定的) 承诺问题 $(S_{\text{yes}}, S_{\text{no}})$ 中, 可以只考虑 S_{yes} 的成员判定问题, 这意味着, 违背承诺的实例被当作是“否”实例。

(2) 考虑作为满足承诺对象的有效编码的串。即固定任意一个满足承诺的串 x_0 , 我们考虑将所有违背承诺的串都当作是 x_0 。在带承诺 P 的搜索问题 R 中, 这意味着要考虑 (标准) 搜索问题 R , 其中 R 被修改成为: 对任意 $x \notin P$, $R(x) = R(x_0)$ 。类似地, 在 (判定) 承诺问题 $(S_{\text{yes}}, S_{\text{no}})$ 中, 考虑的是 S_{yes} 的成员判定问题 (假设 $x_0 \in S_{\text{no}}$, 否则就考虑 $\{0, 1\}^* \setminus S_{\text{no}}$ 的成员判定问题)。

我们强调, 当承诺可以被高效识别时, 上述的习惯做法 (或修改) 不影响相关 (搜索或判定) 问题的复杂性。也就是说, 不考虑原始的承诺问题, 而是考虑一个 (无承诺的搜索或判定) 问题, 而它与原始的承诺问题在计算上等价。因此, 在某种意义上, 这种习惯做法并没有丢失任何东西。另一方面, 即使当两类问题在计算上等价时, 使用一种能对两者进行区分的定义也是有用的 (我们的确区分了不同的 NP-完全问题, 虽然它们在计算上等价)。从概念上考虑, 最关键的是 (以下将讨论的) 承诺不能被高效识别的情况。

当承诺无法被高效识别时, 上述由承诺问题向计算上等价的标准 (搜索和判定) 问题的

转化并不一定保持了问题的复杂度。此时，不可避免地要使用承诺问题的术语。例如，考虑判定一个哈密顿图是否为 3-可着色图，表面上看，这个问题可能与判定一个给定图既是哈密顿图又是 3-可着色图的问题在复杂性上有根本的不同。

与上述思想相反，我们采用了重点讨论标准判定和搜索问题的习惯。也就是说，本书中讨论的所有复杂性类都被默认为针对着标准的判定和搜索问题，如果有时需要讨论承诺问题，则将明确指出。这些例外出现在 2.4.2 节、6.1.2 节、6.2.3 节及附录 10.1 中。

2.4.2 NP 问题的最优搜索算法

我们实际上指的是 PC 中任意关系的公正搜索问题。回顾 PC 类是指其解的正确性能被高效验证的搜索问题类（见定义 2.3），而公正搜索问题是指向解题者承诺给定实例一定有解的搜索问题（见定义 2.30）。

我们认为存在最优的算法来解决 PC 中任意关系的公正搜索问题。进一步地，我们可以明确提出这样的算法，并在很强的意义上证明该算法是最优的：对任意能解公正搜索问题 $R \in PC$ 的算法，我们所提出的解同一问题的算法最多慢一个常数因子（在这里忽略多项式中固定的加法项，当问题无法在多项式时间内求解时可以去掉这些项）。当然，我们并不知道上述最优算法的时间复杂度（如果知道的话就能解 P-vs-NP 问题了）。事实上，P-vs-NP 问题可归结为判定某个明确定义的算法的时间复杂度（即定理 2.33 中所说的最优算法）。

定理 2.33 对任意二元关系 $R \in PC$ ，存在算法 A 满足以下两个条件。

- (1) A 能解 R 的公正搜索问题。
- (2) 存在多项式 p 使得对任意解 R 的公正搜索问题的算法 A' ，以及任意 $x \in S_R$ （即对任意 $(x, y) \in R$ ），有 $t_A(x) = O(t_{A'}(x) + p(|x|))$ ，其中 $t_A(x)$ （或 $t_{A'}(x)$ ）表示 A （或 A' ）对于输入 x 的运行步数。

有趣的是，在证明 A 的最优性时，并不知道其（最优）运行时间。进一步地，最优性的断言是“逐点的”（Point-wise）（即针对任意输入）而非“全局的”（Global）（即将（最坏情况下）时间复杂度作为输入长度的函数）。

我们强调在复杂度的 O -记号中，隐藏的常量只依赖于 A' ，但是在以下的证明中，这种依赖性为算法 A' 描述长度的指数函数（还不知道是否可以得到更好的关系）。这种依赖性以及其中所体现的思想其实是该结论的一个缺陷，否则，该结论绝对完美得令人惊奇。另一个弱点是算法 A 的最优性只针对着那些有解的输入（即属于 S_R 的输入）。最后我们指出，上述定理只适用于那些可以模拟具有常数负载的机器的给定步数的计算模型。我们认为，在多数计算模型下，这种模拟过程的负载最多是计算步数的多项式一对数（Poly-logarithmic），此时有 $t_A(x) = \tilde{O}(t_{A'}(x) + p(|x|))$ 。

证明概要：对于给定的 R ，设 M 为判定 R 成员的多项式时间算法， p 为 M 运行时间的多项式上限（是输入对中第一个元素长度的函数）。利用 M 可以构造如下解 R 的公正搜索问题的算法 A 。对于输入 x ，算法 A “并行地”模拟所有可能的搜索算法，检查每个算法提供的解（利用 M ），并当识别出正确解时停机。事实上，模拟的大多数算法完全与搜索无关，但是利用 M ，可以屏蔽这些算法提供的错误解，并一旦得到正确解时便输出该解。

由于可能存在无限多种算法，所以这里所说的“并行模拟所有可能算法”的含义并不明确。其实这句话的意思是以不同的“比例”来模拟这些算法，使得无限个比例之和为 1（或其他常量）。特别地，我们会以比例 $\frac{1}{(i+1)^2}$ 模拟第 i 个可能的算法，也就是说，对于该算法的

每一步，需要 $(i+1)^2$ 步来模拟（对所有算法都这样执行）。注意：如果要直接实现这个思想，将需要巨大的计算负载，因为要频繁地从对一个机器的模拟转向对另一个的模拟。因此，我们提出另一种迭代的实现方案。

在第 j 次迭代中，对 $i=1, 2, \dots, 2^{j/2}-1$ ，算法 A 模拟第 i 个机器的 $2^j/(i+1)^2$ 步（其中这些机器按其描述的词典序排列）。每次模拟在一个时间块内实现，因此在不同模拟间转换所需的负载可以忽略不计（与模拟的总步数相比）。如果一个或多个模拟（对输入 x ）停机并输出 y ，则算法 A 对输入 (x, y) 调用 M ，并当且仅当 $M(x, y)=1$ 时输出 y 。另外，对于由候选算法输出的解的验证过程也将被模拟，代价是要将其运算步数计入总负载中（换句话说，我们为每个算法增加一个权威性过程（即 M ），它可以检查算法输出解的正确性）。

由构造可以看出，只要 $A(x)$ 输出一个串 y （即 $y \neq \perp$ ），则必有 $(x, y) \in R$ 。为证明 A 是最优的，考虑任意一个解公正搜索问题 R 的算法 A' 。要证明 A 不比 A' 慢很多。直观上看，这是由于 A 的负载是由对其他算法（除 A' 之外）的模拟而引起的，但是模拟中“浪费”的总步数（由于这些算法）与对 A' 的模拟比例成反比，这又反过来与 A' 的描述长度呈指数关系。这里的巧妙之处在于 A' 是固定的，所以其描述的长度为常量，具体细节如下。

对任意 x ，令 $t'(x)$ 表示 A' 对输入 x 的运算步数，而 $t'(x)$ 还可表示 $M(x, \cdot)$ 的运行时间，即 $t'(x) = t_{A'}(x) + p(|x|)$ ，其中 $t_{A'}(x)$ 为 $A'(x)$ 的运算步数。因此，如果 $2^j / (2^{|A'|+1})^2 \geq t'(x)$ ，则模拟 A' 对于输入 x 的运算的 $t'(x)$ 步“包含”于 A 的第 j 次迭代中，其中 $|A'|$ 表示 A' 的描述长度（事实上，我们利用了这样一个事实，即对算法的模拟依词典序进行，且在 A' 之前最多有 $2^{|A'|+1} - 2$ 个算法）。因此，对于输入 x ，算法 A 最多在 $j_{A'}(x)$ 次迭代后停机，其中 $j_{A'}(x) = 2(|A'|+1) + \log_2(t_{A'}(x) + p(|x|))$ ，模拟的总步数为

$$t(x) \stackrel{\text{def}}{=} \sum_{j=1}^{j_{A'}(x)} \sum_{i=1}^{2^{j/2}-1} \frac{2^j}{(i+1)^2} < 2^{j_{A'}(x)+1} = 2^{2|A'|+3} \cdot (t_{A'}(x) + p(|x|))$$

问题在于模拟这些步骤所需的时间取决于具体的计算模型。在许多计算模型中，一个机器的 t 步执行可能需要另一个机器用 $\tilde{O}(t)$ 步来模拟。而在某些模型中，这个模拟过程的负载甚至是常数。定理得证。 $\square$

说明：由构造，上述的算法 A 对于输入 $x \notin S_R$ 不停机。这可以修改，令 A 直接模拟穷举搜索，当且仅当穷举搜索指出当前输入无解时，停机并输出 $\perp$ 。这个额外的模拟可以与其他模拟过程并行进行（即其他模拟每执行一步，这个额外模拟的比例也是一步）。

2.4.3 coNP 类及其与 NP 的交集

通过在复杂性类（判定问题）的名称前加上“co”，可以定义该类的补集，即

$$\text{co}C \stackrel{\text{def}}{=} \{\{0,1\}^* \setminus S : S \in C\} \quad (2.4)$$

特别地, $\text{coNP} = \{\{0,1\}^* \setminus S : S \in \text{NP}\}$ 为 NP 类中集合的补集所构成的类。

回顾 NP 类中的集合以其证据关系为特征, 满足 $x \in S$ 当且仅当存在足够多的 NP-证据, 从而其补集中包含了所有无 NP-证据的实例 (即 $x \in \{0,1\}^* \setminus S$, 如果不存在 x 属于 S 的任何证据)。例如, $\text{SAT} \in \text{NP}$ 蕴含着不可满足的 CNF 范式集合属于 coNP 。类似地, 不可 3-着色的图组成的集合也属于 coNP (后面章节中将提到, 普遍认为这些集合不属于 NP)。

另一个关于 coNP 的观点是由 PC 中的搜索问题得到的。前面讲过, 对任意的 $R \in \text{PC}$, 有解的实例集 (即 $S_R = \{x : \exists y, \text{使得}(x, y) \in R\}$) 属于 NP 。从而, 无解的实例集 (即 $\{0,1\}^* \setminus S_R = \{x : \forall y, (x, y) \notin R\}$) 属于 coNP 。

普遍认为 $\text{NP} \neq \text{coNP}$ (这表示 NP 对于求补运算不封闭)。事实上, 这一假设蕴含了 $\text{P} \neq \text{NP}$ (因为 P 对于求补是封闭的)。 $\text{NP} \neq \text{coNP}$ 的假设意味着 coNP 中某些集合没有 NP-证明系统 (因为 NP 是指那些有 NP-证明系统的集合)。下面我们将证明, 在这个假设下, NP-完全集合的补集没有 NP-证明系统, 例如, 不存在 NP-证明系统来证明一个给定的 CNF 范式是不可满足的。我们首先在标准意义下证明 NP-完全性 (即利用定义 2.17 中的 Karp-归约)。

命题 2.34 假设 $\text{NP} \neq \text{coNP}$, 并令 $S \in \text{NP}$ 使得 NP 中任意集合可以 Karp-归约到 S , 则 $\bar{S} \stackrel{\text{def}}{=} \{0,1\}^* \setminus S$ 不属于 NP 。

证明概要: 首先观察到一个事实, 即 NP 中任意集合可以 Karp-归约到 S , 这蕴含了 coNP 中任意集合可以 Karp-归约到 $\bar{S}$ 。然后, 利用一个断言, 即如果 S' 属于 NP , 则任意一个可以 Karp-归约到 S' 的集合也属于 NP 。将这个断言应用于 $S' = \bar{S}$, 则可得到 $\bar{S} \in \text{NP}$ 蕴含了 $\text{coNP} \subseteq \text{NP}$, 这又蕴含了 $\text{NP} = \text{coNP}$, 从而与假设矛盾。

下面证明上述断言。也就是说, 要证明如果 S' 有一个 NP-证明系统, 且 S'' 可以 Karp-归约为 S' , 则 S'' 也有 NP-证明系统。令 V' 为与 S' 相关的验证程序, 并令 f 为 S'' 到 S' 的 Karp-归约, 则定义验证程序 V'' (验证 S'' 的成员) 为 $V''(x, y) = V'(f(x), y)$, 就是说, $f(x) \in S'$ 的任意 NP-证据都可作为 $x \in S''$ 的 NP-证据 (且这些就是 $x \in S''$ 的所有 NP-证据)。这可能不是一种“自然的”证明系统 (对于 S''), 但它确实是 S'' 的 NP-证明系统。□

假设 $\text{NP} \neq \text{coNP}$, 则命题 2.34 蕴含了 $\text{NP} \cap \text{coNP}$ 中的集合在 Karp-归约下不可能是 NP-完全的。考虑到关于 Karp-归约的其他限制 (如见习题 2.7), 读者可能会疑惑 $\text{NP} \cap \text{coNP}$ 中的集合不具备 NP-完全性是否由于使用了归约的受限版本 (即 Karp-归约)。以下定理指出, 情况并非如此: 即使是在一般的归约下 (即 Cook-归约), NP 中某些集合也不能归约为 $\text{NP} \cap \text{coNP}$ 中的集合。

定理 2.35 若 NP 中任意集合可以 Cook-归约为 $\text{NP} \cap \text{coNP}$ 中某个集合, 则 $\text{NP} = \text{coNP}$ 。

特别地, 假设 $\text{NP} \neq \text{coNP}$, 则 $\text{NP} \cap \text{coNP}$ 中不存在 NP-完全集合, 即使 NP-完全性是用 Cook-归约定义的。由于 $\text{NP} \cap \text{coNP}$ 被假设为是 P 的超集 (Superset), 从而 (在假设

$\mathcal{NP} \neq \text{co}\mathcal{NP}$ 时) $\mathcal{NP}$ 中有一些判定问题既不属于 $\mathcal{P}$ 也不是 NP-困难的(即 $\mathcal{NP} \cap \text{co}\mathcal{NP} \setminus \mathcal{P}$ 中的判定问题)。我们强调定理 2.35 针对的是标准判定问题而非承诺问题(见 2.4.1 节及习题 2.36)。

证明: 类似于命题 2.34, 本定理的证明归结为如果 S 可以 Cook-归约到 $\mathcal{NP} \cap \text{co}\mathcal{NP}$ 中的某个集合, 则 $S \in \mathcal{NP} \cap \text{co}\mathcal{NP}$ 。利用这一断言, 定理的假设中蕴含了 $\mathcal{NP} \subseteq \mathcal{NP} \cap \text{co}\mathcal{NP}$, 这又蕴含着 $\mathcal{NP} \subseteq \text{co}\mathcal{NP}$ 以及 $\mathcal{NP} = \text{co}\mathcal{NP}$ (见习题 2.37)。

给定任意 S 和 $S' \in \mathcal{NP} \cap \text{co}\mathcal{NP}$, 且 S 可以 Cook-归约为 S' , 我们证明 $S \in \mathcal{NP}$ (类似地可证明 $S \in \text{co}\mathcal{NP}$)。^① 设 M 为将 S 归约到 S' 的预言机, 即对于输入 x , 机器 M 发出询问, 并判断是否接受 x , 如果对所有询问的应答都是根据 S' 做出的, 则其判定是正确的。为证明 $S \in \mathcal{NP}$, 我们构造一个关于 S 的 NP-证明系统。该证明系统(或更确切地, 其验证程序), 记作 V , 在以下两个条件成立时, 接受形如 $(x, ((z_1, \sigma_1, \omega_1), \dots, (z_i, \sigma_i, \omega_i)))$ 的对。

(1) 对于输入 x , 机器 M 在发出询问 $z_1, z_2, \dots, z_i$ 并得到相应回答 $\sigma_1, \sigma_2, \dots, \sigma_i$ 之后, 接受 x 。

即对于输入 x , 机器 V 在得到前 $i-1$ 个询问的应答 $\sigma_1, \sigma_2, \dots, \sigma_{i-1}$ 之后, 检查 M 发出的第 i 个询问是否等于 z_i 。另外, 对于输入 x , V 在收到应答 $\sigma_1, \sigma_2, \dots, \sigma_i$ 之后, 检查 M 是否停机并输出 1 (表示接受)。

注意: V 并不能对 S' 进行预言访问。程序 V 对于 $M(x)$ 的模拟是通过对于每个 i , 利用比特 σ_i 来回答 $M(x)$ 发出的第 i 个询问(条件是将 V 作为其输入的一部分)。这些应答的正确性将由 V 独立地验证(即见以下的条件(2))。

(2) 对每个 i , 如果 $\sigma_i = 1$, 则 ω_i 是 $z_i \in S'$ 的 NP-证据, 而当 $\sigma_i = 0$ 时, ω_i 是 $z_i \in \{0, 1\}^* \setminus S'$ 的 NP-证据。

因此, 如果条件(2)成立, 则每个 σ_i 都表示 z_i 关于 S' 的正确状态(即 $\sigma_i = 1$ 当且仅当 $z_i \in S'$)。

我们强调, 在这里利用了一个事实, 即 S' 和 $\bar{S}' = \{0, 1\}^* \setminus S'$ 都具有 NP-证明系统, 且针对着相应的 NP-证据。

注意: V 实际上是 S 的一个 NP-证明系统。首先, 相应证据的长度受归约运行时间(以及为各种询问提供的 NP-证据的长度)限制。还要注意, V 运行于多项式时间(即验证条件(1)要求在利用 σ_i 模拟预言时, 要模拟 M 对于输入 x 的多项式时间执行, 而条件(2)的验证要调用相关的 NP-证明系统)。最后, 观察到 $x \in S$ 当且仅当存在序列 $y = ((z_1, \sigma_1, \omega_1), \dots, (z_i, \sigma_i, \omega_i))$ 满足 $V(x, y) = 1$ 。特别地, 只有当 y 中包含一个有效的询问序列, 以及根据 M 对输入 x 的计算, 在能访问 S' 时得到的应答序列, 且 M 基于该序列接受 x 时, 才有 $V(x, y) = 1$ 。■

全局观点——总结

在 $\mathcal{P} \neq \mathcal{NP}$ 的猜想之上, 我们还提出另外两个猜想(其中显然蕴含了 $\mathcal{P} \neq \mathcal{NP}$)。

(1) $\mathcal{NP} \neq \text{co}\mathcal{NP}$ (等价地, $\mathcal{NP} \cap \text{co}\mathcal{NP} \neq \mathcal{NP}$)。这个猜想等价于猜想 CNF 范式没

① 另外, 我们应用如下关于 $\bar{S}' = \{0, 1\}^* \setminus S'$ 的论据来证明 $S \in \text{co}\mathcal{NP}$, 并注意到 $\bar{S}'$ 可以 Cook-归约为 S' (通过 S , 或 $\bar{S}'$ 可以 Cook-归约为 $\{0, 1\}^* \setminus S' \in \mathcal{NP} \cap \text{co}\mathcal{NP}$)。

视为包含了“大范围的重要计算问题（不属于 P 类）”。

独立于这些研究，同一时代在苏联，Levin 证明了“通用搜索问题”的存在性（其中通用的意思就是 NP-完全性）。Levin 工作^[152]的出发点是其对于“perebor”猜想的兴趣，该猜想称某些具有可高效验证解的搜索问题（即 PC 中的问题）在本质上只能用穷举搜索法求解。Levin 强调，相关问题的时间复杂度之间的关系（对时间复杂度的任意增长率）中可能存在的多项式时间归约，他证明了 6 个“经典搜索问题”的 NP-完全性，并声称所采用的证明方法对于“许多其他重要的搜索问题”可以很容易地得到类似结果。

有趣的是，虽然人们在不同程度上接受了 Cook^[58]、Karp^[138]和 Levin^[152]的工作，但是同时代中没有人能意识到这些发现的深远意义，以及所提出问题（即 P-vs-NP 问题）的困难程度。自 20 世纪 70 年代以来，这一事实在各个方面都比较明显，并且也解释了这一代研究者遇到的挫折，他们希望穷其一生的精力（如果不能在几年内完成）来解决 P-vs-NP 问题。当然，作者的观点是这种期望并非完全合理，人们实际上应该抱有相反的期望。

我们指出，NP-完全性理论的 3 篇“奠基性论文”（即 Cook^[58]、Karp^[138]、Levin^[152]）中使用了本章介绍的 3 种不同类型的归约。具体而言，Cook 使用了一般意义上的多项式时间归约^[58]，通常称为 Cook-归约（定义 2.9）。Karp-归约的概念（定义 2.11）来自于 Karp 的文章^[138]，而 Levin 的文章^[152]中首次将其扩展到搜索问题（即定义 2.12）。值得一提的是，Levin 的工作是用搜索问题陈述的，而不像 Cook 和 Karp 的工作，考虑的是判定问题。

2.3.3.2 节中的归约并非最初的归约。尤其是在证明独立集问题的 NP-完全性（即命题 2.26）时使用的归约来自于参考文献[74]（还可见于习题 9.18）。相反，2.3.3.1 节中的归约仅仅是参考文献[58]中原始归约的重述。 NP 的两种定义的等价性（即定理 2.8）在参考文献[138]中得到证明。

既非 P 也非 NP-完全的 NP-集的存在性（即定理 2.28）由 Ladher^[149]证明。定理 2.35 由 Selman^[198]证明，而 NP-关系的最优搜索算法的存在性（即定理 2.33）由 Levin^[152]证明。（有趣的是，后一个结论也来自于那篇 Levin 独立于 Cook 和 Karp，发现了 NP-完全性的论文）。承诺问题的概念由 Even、Selman 和 Yacobi 第一次明确引入^[72]，参考文献[94]中对其丰富应用进行了综述。

我们指出，用于证明自然的 NP-完全性结论时使用的标准归约有几个附加性质，或可以被修改为具有这些性质。这些性质包括可以对问题实例的解按照归约的方向进行高效转换（见习题 2.28）、解的数量的保持（见习题 2.29）、对数空间算法可计算性（见 5.2.2 节），以及多项式时间可求逆（见习题 2.30）。还要指出一个事实，所有已知的 NP-完全集都是（高效）同构的（见习题 2.13）。

习 题

2.1 PF 中包含不属于 PC 的问题：证明 PF 中包含某些（非自然的）不属于 PC 的问题。

提示：考虑关系 $R = \{(x, 1) : x \in \{0, 1\}^*\} \cup \{(x, 0) : x \in S\}$ ，其中 S 为某个不可判定的集合。

注意： R 是两个二元关系的不相交并集，这两个关系记作 R_1 和 R_2 ，其中 R_1 属于 PF 而 R_2 不属于 PC 。另外，对任意 x ，有 $R_1(x) \neq \emptyset$ 。

2.2 证明任意 $S \in NP$ 有许多个不同的 NP-证明系统（即验证程序 $V_1, V_2, \dots$ 使得对 $i \neq j$ ，

$V_i(x, y) = 1$ 并不蕴含 $V_j(x, y) = 1$ 。

提示: 对如定义 2.5 中的 V 和 p , 如果 $|y| = p(|x|) + i$, 则定义 $V_i(x, y) = 1$, 从而存在 y 的前缀 y' 使 $V(x, y') = 1$ 。

2.3 根据可在多项式时间内判定素性的事实, 并假设不存在多项式时间的整数分解算法, 对于合数集, 提出两种“自然的但本质上不同的”NP-证明系统。

提示: 对于合数集, 考虑如下的验证程序 V_1 和 V_2 。令 $V_1(n, y) = 1$ 当且仅当 $y = n$ 且 n 不是素数, 而令 $V_2(n, m) = 1$ 当且仅当 m 为 n 的非平凡因子。证明易找到关于 V_1 的有效证明, 而寻找关于 V_2 的有效证明是困难的。

2.4 关于定义 2.7, 证明如果 S 可由某个时间复杂度为 t 的非确定机器接受, 则其也可由时间复杂度为 $O(t)$ 的非确定机器接受, 且后者的迁移函数将每个可能的符号—状态对恰好映射为两个三元组。

2.5 验证 Cook-归约的以下性质。

(1) 如果 Π 可以 Cook-归约为 Π' , 且 Π' 可在多项式时间内求解, 则 Π 也可在多项式时间内求解。

(2) Cook-归约具有传递性(即如果 Π 可以 Cook-归约为 Π' , 而 Π' 可以 Cook-归约为 Π'' , 则 Π 可以 Cook-归约为 Π'')。

(3) 如果 Π 可在多项式时间内求解, 则它可以 Cook-归约为任意问题 Π' 。

作为性质(3)的延续, 证明问题 Π 可在多项式时间内求解当且仅当它可以 Cook-归约为一个平凡问题(如空集的成员判定问题)。

2.6 证明 Karp-归约(及 Levin-归约)具有传递性。

2.7 证明某些判定问题不能 Karp-归约为其补问题(如空集不能 Karp-归约为 $\{0, 1\}^*$)。

关于模糊性一个常见的习题是证明 $\mathcal{P}$ 中任意一个判定问题可以 Karp-归约为任意一个非平凡的判定问题。关于集合 S 的判定问题称为非平凡的, 如果 $S \neq \emptyset$ 且 $S \neq \{0, 1\}^*$, 从而有 $\mathcal{P}$ 中每个非平凡集合可以 Karp-归约为其补集。

2.8 (搜索问题到优化问题的归约) 对任意多项式界限的关系 R (或 $R \in PC$), 构造一个函数 f (或多项式时间可计算的函数 f), 使得 R 的搜索问题在计算上与这样一个问题等价: 给定 (x, v) , 求 $y \in \{0, 1\}^{\text{poly}(|x|)}$, 使得 $f(x, y) \geq v$ 。

提示: 利用一个布尔函数。

2.9 (二分查找法) 证明利用 l 个形如“是否 $z \geq v$ ”的二元询问, 可以判定整数 z 是否位于区间 $[0, 2^l - 1]$ 中。

提示: 考虑对某个已知包含 z 的区间进行重复的折半。

2.10 证明: 如果 $R \in PC \setminus \mathcal{PF}$ 是自归约的, 则相应的 Cook-归约要向 S_R 发出多于对数次数的询问。更一般地, 证明如果 $R \in PC \setminus \mathcal{PF}$ 可以 Cook-归约为任意的判定问题, 则该归约中至少需要对数次数询问。

提示: 注意, 可以通过尝试所有的可能性来模拟预言应答, 从而预言机输出的正确性可以被高效验证。

2.11 图的 3-可着色性的标准搜索问题是指对于给定图, 求一种 3-着色方案。证明这个问题是自归约的。

提示：通过向图的 3-可着色性的判定问题发出足够多的预言询问来不断延长图的当前 3-着色方案前缀（具体地，将问题 “ $(\chi_1, \chi_2, \dots, \chi_t) \in \{1, 2, 3\}^t$ 是否是图 G 的 3-着色方案前缀” 编码为关于某个辅助图 G' 的 3-可着色性的询问）。^①

2.12 图的同构性的标准搜索问题是指求一对给定图之间的同构映射，证明该问题是自归约的。

提示：通过向图同构的判定问题发出足够多的预言询问来不断延长两个 N -顶点图之间当前同构映射的前缀（具体地，将问题 “ $(\pi_1, \pi_2, \dots, \pi_t) \in [N]^t$ 是否为图 $G_2 = ([N], E_2)$ 与 $G_1 = ([N], E_1)$ 之间同构映射的前缀” 编码为两个辅助图 G'_1 与 G'_2 是否同构的询问）。^②

2.13 （下行自归约性）称集合 S 为下行自归约的，如果存在一种 S 到其自身的 Cook-归约，其中发出的询问长度小于归约的输入长度（即如果对于输入 x ，归约发出的询问 q 满足 $|q| < |x|$ ）。^③

（1）证明在一种自然的 CNF 范式编码下，SAT 是下行自归约的。注意：编码应具有这样一种性质，对范式中一个变量的实例化将得到一个更短的范式。

一个更难的习题是证明在对图的某种合理编码方式下，图的 3-可着色性是下行自归约的。注意：必须仔细地选择编码方式（如果该编码将工作在类似于习题 2.11 的过程中时）。

（2）假设 S 是下行自归约的，且归约的输出是预言应答的析取（注意：SAT 正是这种情况）。证明，此时 S 具有自归约证据关系 $R \in \mathcal{PC}$ 的特征（即 $S = \{x : R(x) \neq \emptyset\}$ ）（即 R 的搜索问题可以 Cook-归约为 S ）。当然，此时必有 $S \in \mathcal{NP}$ 。

提示：如果 $x_i \in S \cap \{0, 1\}^{O(i)}$ ，且对任意 $i \in \{0, 1, \dots, t-1\}$ ，自归约对输入 x_i 发出一系列包含 x_{i+1} 的询问，则向 R 中加入 $(x_0, \langle x_1, x_2, \dots, x_t \rangle)$ 。证明，实际上 $R \in \mathcal{PC}$ 且 $S = \{x : R(x) \neq \emptyset\}$ 。

注意：下行自归约性的概念可以以某种自然的方式加以推广。例如，当 S 可以通过 Karp-归约而与某个下行自归约集（在上述的严格意义下）在计算上等价时，也可以称 S 是下行自归约的。注意此时第 2 部分仍成立。

2.14 （不可自归约的 NP 问题）

（1）假设 $\mathcal{P} \neq \mathcal{NP} \cap \text{coNP}$ ，证明 $\mathcal{PC}$ 中存在非自归约的搜索问题 R 。

提示：给定 $S \in \mathcal{NP} \cap \text{coNP} \setminus \mathcal{P}$ ，构造关系 $R_1, R_2 \in \mathcal{PC}$ ，使得 $S = \{x : R_1(x) \neq \emptyset\} = \{x : R_2(x) = \emptyset\}$ ，然后考虑关系 $R = \{(x, 1y) : (x, y) \in R_1\} \cup \{(x, 0y) : (x, y) \in R_2\}$ ，并证明 $R \notin \mathcal{PF}$ 但 $S_R = \{0, 1\}^*$ 。

（2）证明：如果搜索问题 R 是不可自归约的，则① $R \notin \mathcal{PF}$ ；② 集合 $S'_R = \{(x, y') : \exists y'', s.t. (x, y'y'') \in R\}$ 不能 Cook-归约为 $S_R = \{x : \exists y, s.t. (x, y) \in R\}$ 。

2.15 （对 $\mathcal{PC}$ 和 $\mathcal{PF}$ 的任意解的前缀进行扩展）设 $\mathcal{P} \neq \mathcal{NP}$ ，求 $\mathcal{PC} \cap \mathcal{PF}$ 中的搜索问题 R ，使得判定 S'_R 不能归约为 R 的搜索问题。

① 注意：只需检查图 G 是否有一种 3-着色方案，其中由 $(\chi_1, \chi_2, \dots, \chi_t)$ 诱导出的相等和不等关系都成立。这可以通过足够多的分图（如用相应顶点间的边来实现不等关系，而用一个包含了相关顶点以及辅助顶点的子图来实现相等关系）。在习题 2.13 的第一部分，将两个顶点相结合，可以更好地实现相等关系。

② 这可以通过将足够多的分图与期望构成（同构）映射的顶点对相关联。例如，可以将第 i 对中的顶点关联到一个辅助的，包含 $(N+i)$ 个顶点的星图中。

③ 注意：对于某些实例，归约根本不发出询问（这避免了由于实例非常短而使定义失效的可能）。

提示: 考虑关系 $R = \{(x, 0x) : x \in \{0, 1\}^*\} \cup \{(x, 1y) : (x, y) \in R'\}$, 其中 R' 为 $\mathcal{PC} \setminus \mathcal{PF}$ 中任意一个关系, 证明 $R \in \mathcal{PF}$ 但 $S_R' \notin \mathcal{P}$ 。

2.16 续习题 2.14, 求 $\mathcal{PC}$ 中一种自然的搜索问题 R , 满足如果整数分解是困难的, 则 R (以及 S_R') 的搜索问题不可归约为 S_R 。

提示: 考虑关系 R , 满足当 Q 为整数 N 的非平凡因子时, $(N, Q) \in R$ 。利用素数集属于 $\mathcal{P}$ 这一事实可证。

2.17 续习题 2.14 及习题 2.16, 证明在恰当的假设下, 存在关系 $R_1, R_2 \in \mathcal{PC}$, 其中蕴含着相同的判定问题 (即 $\{x : R_1(x) \neq \phi\} = \{x : R_2(x) \neq \phi\}$), 且 R_1 是自归约的, 而 R_2 不是。

2.18 给出定理 2.16 的另一种证明, 其中不涉及集合 $S_R' = \{(x, y') : \exists y'', s.t. (x, y'y'') \in R\}$ (提示: 利用命题 2.15)。

提示: 将 R 的搜索问题归约为 R_{SAT} 的搜索问题, 再将 R_{SAT} 归约为 SAT, 最后将 SAT 归约为 S_R 。证明这三种归约的存在性。

2.19 证明有界停机问题 (Bounded Halting) 和有界不停机问题 (Bounded Non-Halting) 是 NP-完全的, 这两个问题定义如下: 问题的实例中包含一个对 (M, l') , 其中 M 为图灵机, t 为整数。有界停机 (或有界不停机) 问题的判定形式是指确定是否存在一个输入 (长度至多为 t), 使 M 在 t 步内停机 (或不停机), 而搜索形式是找到这样一个输入。

提示: 或者修改定理 2.19 的证明, 或者构造 R_u 的搜索问题到有界停机 (或不停机) 问题的归约 (事实上, 对于有界停机的情况, 本题更加直接)。

2.20 在定理 2.21 的证明中, 我们声称, 机器 M 的“配置列表”中每一项的值都由其上一行中的三项决定 (图 2.1)。构造一个函数 $f_M : \Gamma^3 \rightarrow \Gamma$, 其中 $\Gamma = \Sigma \times (Q \cup \{\perp\})$, 可以例证这个断言。

提示: 例如, 对任意 $\sigma_1, \sigma_2, \sigma_3 \in \Sigma$, 有 $f_M((\sigma_1, \perp), (\sigma_2, \perp), (\sigma_3, \perp)) = (\sigma_2, \perp)$ 。更有趣的是, 如果 M 的迁移函数将 (σ, q) 映射为 $(\tau, p, +1)$, 则对任意 $\sigma_1, \sigma_2, \sigma_3 \in Q$, 有 $f_M((\sigma, q), (\sigma_2, \perp), (\sigma_3, \perp)) = (\sigma_2, p)$ 且 $f_M((\sigma_1, \perp), (\sigma, q), (\sigma_3, \perp)) = (\tau, \perp)$ 。

2.21 构造并分析从 SAT 到 3SAT 的归约。

提示: 对于子句 C , 考虑一组辅助变量, 其中第 i 个变量指示着是否前 i 个文字为可满足的, 并用一个 3CNF 范式来代替 C , 其中的变量包括 C 中的初始变量以及所有的辅助变量。例如, 子句 $\vee_{i=1}^t x_i$ 被逻辑上等价于表达式 $(y_2 \equiv (x_1 \vee x_2)), (y_i \equiv (y_{i-1} \vee x_i)), (i=3, 4, \dots, t)$ 及 y_t , 的 3CNF 范式的合取所代替。我们要指出, 这并非一种标准归约, 但是它在概念上更吸引人。^①

2.22 (2SAT 可以高效求解) 与习题 2.21 相反, 证明 2SAT (即 2CNF 范式的可满足性) 属于 $\mathcal{P}$ 类。

提示: 考虑如下对 CNF 范式的“强制性过程”。如果范式中包含一个独立 (Singleton) 子句 (即只有一个文字的子句), 则可以为相应的变量分配唯一的值使该子句满足, 从而简化该范式 (可能会得到一个常数范式, 或者为“真”, 或者为“假”)。这个过程可以重复下去, 直至得到一个常数或只包含非独立子句的范式。注意: 公式 ϕ 是可满足的当且仅当将上述过

^① 在标准归约中, 子句 $\vee_{i=1}^t x_i$ 被 3CNF 范式 $(x_1 \vee x_2 \vee z_2), ((\neg z_{i-1}) \vee x_i \vee z_i)$ 的合取所代替, 其中 $i=3, 4, \dots, t$ 及 $\neg z_t$ 。

程作用于 ϕ 后得到的公式是可满足的。考虑以下算法，可以解与 2SAT 相关联的搜索问题：

(1) 选择 ϕ 中任意一个变量。对每个 $\sigma \in \{0,1\}$ ，将为该变量赋值 σ 之后得到的公式记作 ϕ_σ 。

(2) 如果对某个 $\sigma \in \{0,1\}$ ，将上述强制性过程作用于 ϕ_σ 之后得到一个（非常数的）2CNF 范式 ϕ' ，则令 $\phi \leftarrow \phi'$ 并转到第一步（当这一事件对 $\sigma \in \{0,1\}$ 的两种情况均成立时，视为只对一个 σ 成立，即此时我们可以任选一个 σ ）。

(3) 如果这些赋值中有一个（通过应用上述强制性过程）可以得到常数“真”，则停止，并得到初始公式的可满足性赋值，否则即两种赋值都得到常数“假”，则停止，并断言原始公式是不可满足的。

证明：这个算法的正确性可以归因于在第(2)步中进行的任意一种选择都是非实质性的。事实上，这一结论依赖于只针对 2CNF 范式的情形。

2.23 （整数线性规划） 证明以下问题是 NP-完全的。问题的实例为：给定一组（整系数）线性不等式，判定这组不等式是否有一个整数解。这个判定问题的一个典型实例可能是这样的。

$$\begin{aligned}x + 2y - z &\geq 3 \\ -3x - z &\geq -5 \\ x &\geq 0 \\ -x &\geq -1\end{aligned}$$

提示：将 SAT 归约为该问题。具体地，考虑对于输入 CNF 的算术化过程，将 $\vee$ 运算用加法代替，而将 $\neg x$ 用 $1-x$ 代替。从而，每个子句可以诱导出一个不等式（如子句 $x \vee \neg y$ 被不等式 $x + (1-y) \geq 1$ 所代替，后者又可以简化为 $x - y \geq 2$ ）。为每个变量 x 引入形如 $x \geq 0$ 和 $-x \geq -1$ 的不等式，则可以强制性地得到一种 0-1 解。

2.24 （GF(2) 上线性方程组的最大可满足性） 证明如下问题是 NP-完全的。问题实例中包含 GF(2) 上的一个线性方程组及整数 k ，判断是否存在一种赋值，至少使 k 个方程成立（注意：判断是否存在一种赋值能满足所有方程属于 P 问题）。

提示：利用类似于习题 2.23 的算术化过程，将 3SAT 问题归约为此问题。具体地，将每个含 $t \leq 3$ 个文字的子句用 $2^t - 1$ 个 GF(2) 上的线性方程替换，这些方程对应于这些文字的不同非空子集，并确保其和 (mod 2) 等于 1；例如，子句 $x \vee \neg y$ 用方程 $x + (1-y) = 1$ ， $x = 1$ 和 $1-y = 1$ 替换。将 {false, true} 等同于 $\{0,1\}$ ，证明如果一种布尔赋值 $\bar{v}$ 满足初始子句，则 $\bar{v}$ 恰好能满足 2^{t-1} 个相应的方程；当 $\bar{v}$ 不能满足初始子句时，它将无法满足任何一个相对应的方程。

2.25 （GF(2) 上二次方程组的可满足性） 证明以下问题是 NP-完全的。问题实例包含 GF(2) 上一组二次方程，判断是否存在一种赋值，能满足所有方程。注意这个结论对实数域上的二次方程组也成立（通过一个附加条件，强制性地赋值为 $\{0,1\}$ ）。

提示：先证对于三次方程组，相应问题是 NP-完全的。可由 3SAT 归约，将子句 $x \vee \neg y \vee z$ 映射为方程 $(1-x) \cdot y \cdot (1-z) = 0$ 。然后，引入辅助变量，将 3 次方程简化为 2 次方程，方法是：设实例中的变量为 $x_1, x_2, \dots, x_n$ ，引入辅助变量 $x_{i,j}$ 并增加形如 $x_{i,j} = x_i \cdot x_j$ 的方程。

2.26 （团和独立集） 独立集 (Independent Set) 问题的实例中包含一个对 (G, K) ，其中 G 为图， K 为整数，问图 G 中是否包含一个规模（至少）为 K 的独立集（即一个顶点集，其中任意一对顶点之间都没有边相连）。团 (Clique) 问题是相似的。利用 Karp-归约，证明这两个问题在计算上均等价于顶点覆盖问题。

2.27 (命题 2.26 的另一种证法) 考虑如下由 3SAT 到独立集问题的归约。对于输入的含 m 个子句和 n 个变量的 3CNF 范式 ϕ , 构造一个图 G_ϕ , 包含 m 个三角形 (对应于 m 个子句) 以及将互逆的文字相连的边。即如果 x 是第 j_1 个子句的第 i_1 个文字, 而 $\neg x$ 是在第 j_2 个子句的第 i_2 个文字, 则在第 j_1 个三角形的第 i_1 个顶点与第 j_2 个三角形的第 i_2 个顶点之间画一条边。证明: $\phi \in 3SAT$ 当且仅当 G_ϕ 中含规模为 m 的独立集。

2.28 (标准归约的其他性质) 续正文中的讨论, 考虑如下 Karp-归约的增强形式。 R 到 R' 的归约由三个多项式时间映射 (f, h, g) 组成, 其中 f 为 S_R 到 $S_{R'}$ 的 Karp-归约, 且满足以下两个条件。

(1) 对任意 $(x, y) \in R$, 有 $(f(x), h(x, y)) \in R'$ 。

(2) 对任意 $(f(x), y') \in R'$, 有 $(x, g(x, y')) \in R$ 。

(注意: 上述定义实际上是 Levin 在参考文献[152]中使用的定义, 除了其中对 h 和 g 进行限制, 使其仅依赖于第二个参数之外。)

证明: 这种归约同时蕴含了 Karp-归约和 Levin-归约, 并证明本章中所有的归约都满足这种性质。进一步地, 证明在所有这些情况下, 主要映射 (即 f) 为双射, 且可在多项式时间内求逆。

2.29 (简化的归约) 令 $R, R' \in PC$, f 为从 $S_R = \{x: R(x) \neq \emptyset\}$ 到 $S_{R'} = \{x: R'(x) \neq \emptyset\}$ 的 Karp-归约。称 f (关于 R 和 R') 为简化的归约, 如果对任意 x , 有 $|R(x)| = |R'(f(x))|$ 。检查本章中介绍的每一种归约是否为简化的。对于非简化的归约, 求与其等效的简化归约 (见参考文献[85]中 7.3 节)。

2.30 (多项式时间可逆的归约 (续参考文献[37])) 称集合 S 是可标记的, 如果存在多项式时间 (标记) 算法 M , 满足:

(1) 对任意 $x, \alpha \in \{0, 1\}^*$, 有

① $M(x, \alpha) \in S$ 当且仅当 $x \in S$;

② $|M(x, \alpha)| > |x|$ 。

(2) 存在多项式时间 (标记解除) 算法 D , 满足对任意 $x, \alpha \in \{0, 1\}^*$, 有 $D(M(x, \alpha)) = \alpha$ 。

注意: 所有自然的 NP-集 (如本章中考虑的) 都是可标记的 (如 SAT 问题, 在标记一个公式时, 可以增加可满足的子句, 其中使用特别指明的辅助变量)。证明: 如果 S' 可以 Karp-归约为 S , 且 S 是可标记的, 则 S' 到 S 的 Karp-归约是一种长度增加且多项式时间可求逆的一一映射。^①由此推论出对任意自然的 NP-完全问题 S , $\mathcal{NP}$ 中任意集合可以通过一种长度增加且多项式时间可求逆的一一映射来 Karp-归约到 S 。

提示: 令 f 为 S' 到 S 的 Karp-归约, M 为标记算法。考虑一种映射, 将 x 映射为 $M(f(x), x)$ 。

2.31 (NP-完全集的同构性 (续参考文献[37])) 设 S 和 T 可以相互 Karp-归约, 且归约为长度增加、多项式时间可求逆的一一映射, 分别记作 f 和 g 。利用以下提示证明, S 和 T 是“高效”同构的; 即构造一个多项式时间计算且可逆的一一映射 ϕ , 满足 $T = \phi(S) \stackrel{\text{def}}{=} \{\phi(x): x \in S\}$ 。

① 当给定一个不属于映射的像的串时, 求逆算法返回一个特殊符号。

(1) 令 $F = \{f(x) : x \in \{0,1\}^*\}$, $G = \{g(x) : x \in \{0,1\}^*\}$ 。利用 f (或 g) 保持长度的性质, 证明 F (或 G) 是 $\{0,1\}^*$ 一个适当的子集。证明对任意 $y \in \{0,1\}^*$, 存在唯一的三元组 $(j, x, i) \in \{1,2\} \times \{0,1\}^* \times (\{0\} \cup \mathbb{N})$, 满足下列条件之一。

① $j=1, x \in \overline{G} = \{0,1\}^* \setminus G$ 且 $y = (g \circ f)^i(x)$ 。

② $j=2, x \in \overline{F} = \{0,1\}^* \setminus F$ 且 $y = (g \circ f)^i g(x)$ 。

(在两种情况下, 均有 $h^0(z) = z$, $h^i(z) = h(h^{i-1}(z))$, 且 $(g \circ f)(z) = g(f(z))$ 。提示: 考虑可应用于 y 的最大逆操作序列 $g^{-1}, f^{-1}, g^{-1}, \dots$, 并注意到每次求逆都使当前串长度缩小。)

(2) 令 $U_1 = \{(g \circ f)^i(x) : x \in \overline{G} \wedge i \geq 0\}$, $U_2 = \{(g \circ f)^i(g(x)) : x \in \overline{F} \wedge i \geq 0\}$ 。证明 (U_1, U_2) 构成 $\{0,1\}^*$ 的一种划分。利用 f 和 g 均为长度增长且多项式时间可求逆这一特性, 构造一个多项式时间过程, 能判定集合 U_1 的成员。

对集合 $V_1 = \{(f \circ g)^i(x) : x \in \overline{F} \wedge i \geq 0\}$ 和 $V_2 = \{(f \circ g)^i(f(x)) : x \in \overline{G} \wedge i \geq 0\}$, 证明有同样的结论成立。

(3) 注意到 $U_2 \subseteq G$, 当 $x \in U_1$ 时, 定义 $\phi(x) = f(x)$, 否则, 定义 $\phi(x) = g^{-1}(x)$ 。

① 证明 ϕ 是 S 到 T 的 Karp-归约。

② 注意到 ϕ 将 U_1 映射为 $f(U_1) = \{f(x) : x \in U_1\} = V_2$, 将 U_2 映射为 $g^{-1}(U_2) = \{g^{-1}(x) : x \in U_2\} = V_1$, 证明 ϕ 是单射且是满射。

观察到当 $x \in F(U_1)$ 时, $\phi^{-1}(x) = f^{-1}(x)$, 否则 $\phi^{-1}(x) = g(x)$ 。证明 ϕ^{-1} 是 T 到 S 的 Karp-归约, 从而推断 $\phi(S) = T$ 。

利用习题 2.30, 推断所有自然的 NP-完全集都是同构的。

2.32 证明集合 S 可以 Karp-归约为 $\mathcal{NP}$ 中某个集合当且仅当 S 属于 $\mathcal{NP}$ 。

提示: 见命题 2.34 的证明。

2.33 回顾空集不能 Karp-归约到 $\{0,1\}^*$, 而任意集合都可 Cook-归约为空集的补集。因此, 考虑非平凡集能否 Karp-归约为其补集。称一个集合是非平凡的, 如果它既非空集也不是包含所有的串。进一步地, 由于 $\mathcal{P}$ 中任意非平凡集合可以 Karp-归约为其补集 (见习题 2.7), 我们假设 $\mathcal{P} \neq \mathcal{NP}$ 并主要讨论 $\mathcal{NP} \setminus \mathcal{P}$ 中的集合。

(1) 证明: $\mathcal{NP} = \text{co}\mathcal{NP}$ 蕴含着 $\mathcal{NP} \setminus \mathcal{P}$ 中某些集合可以 Karp-归约为其补集。

(2) 证明: $\mathcal{NP} \neq \text{co}\mathcal{NP}$ 蕴含着 $\mathcal{NP} \setminus \mathcal{P}$ 中某些集合不能 Karp-归约为其补集。

提示: 两部分都可利用 NP-完全集, 第二部分还可利用习题 2.32。

2.34 参照定理 2.28 的证明, 证明函数 f 是无界的 (即对任意 i , 存在整数 n , 使得定理证明中定义的过程在第 n^3 步允许对第 $i+1$ 个机器失效)。

提示: 注意 f 是单调非降的 (因为更多的执行步骤将允许至少在这么多机器中失效)。用反证法, 假设 f 有界, 令 $i = \sup_{n \in \mathbb{N}} \{f(n)\}$, n' 为满足 $f(n') = i$ 的最小整数。如果 i 为奇数, 则由 f 确定的集合 F 是有限的 (因为 $F = \{x : f(|x|) \equiv 1 \pmod{2}\} \supseteq \{x : |x| \geq n'\}$)。此时, 第 $i+1$ 个机器要判断 $S \cap F$ (与 S 在有限个串处不同), 则一定会对某个 x 失效。通过对所需运算步

骤数的计算,并考虑到 x 至多为 $\exp(\text{poly}(|x|))$,这个数当 n 足够大时,一定小于 n^3 ,可以得出矛盾。当 i 为偶数时,可以使用类似方法,此时利用这样一个事实,即 $F \subseteq \{x: |x| < n'\}$ 是有限的,从而 S 到 $S \cap F$ 的归约一定对某个输入 x 失效。

2.35 证明:如果承诺问题 Π 可以 Cook-归约为某个可在多项式时间内求解的承诺问题,则 Π 也可在多项式时间内求解。注意:此时的求解算法一定会对违反承诺的输入停机。

提示:解任意承诺问题的任意多项式时间算法都可以修改为对所有输入停机。

2.36 (coNP 中的 NP-完全承诺问题(续参考文献[72]))考虑承诺问题 xSAT,其实例为 CNF 范式对 (ϕ_1, ϕ_2) 。在“是”实例中, ϕ_1 是可满足的, ϕ_2 不可满足,而“否”实例中, ϕ_1 是不可满足的, ϕ_2 则可满足。

(1) 证明 xSAT 属于类似于 $\mathcal{NP}$ 和 coNP 的承诺问题类的交集。

(2) 证明 $\mathcal{NP}$ 中任意承诺问题可以 Cook-归约为 xSAT。在构造归约时,要令违背承诺的询问得到的应答是任意的。

提示:注意: $\mathcal{NP}$ 类的承诺问题版本可以归约为 SAT,并构造 SAT 到 xSAT 的归约。具体地,利用命题 2.15 的证明思想,证明与 SAT 相关联的搜索问题可以 Cook-归约为 xSAT。即假设已知 τ 为 ϕ 的可满足性赋值的前缀,我们希望为 τ 增加一比特。从而,对每个 $\sigma \in \{0,1\}$,构造一个公式,记作 ϕ_σ' ,方法是令 ϕ 的前 $|\tau|+1$ 个变量取值 $\tau\sigma$ 。向预言机询问 (ϕ_σ', ϕ_0') ,并相应地延长 τ (即当且仅当得到肯定回答时,为 τ 续上 1)。注意:如果 ϕ_1' 和 ϕ_0' 都是可满足的,则在延长时使用什么值并不重要,而如果两者中只有一个是可满足的,则需要根据预言应答来延长 τ 。

(3) 将定理 2.35 的证明应用于上述归约过程时,指出在何处有可能出错。

2.37 对任意类 C ,证明 $C \subseteq \text{co}C$ 当且仅当 $C = \text{co}C$ 。

第3章

P 与 NP 的变形

投出冷眼，看
生，看死，骑
者，向前！

威廉·勃特勒·叶芝，《在本布尔本山下》

本章介绍复杂性类 P 和 NP 的变形。特别要强调 P 的非一致性版本，以及多项式时间层级（Polynomial-time Hierarchy）（这是对 NP 类的一种扩展）。介绍这些变形的原因主要是技术上的考虑，且本章所涉及的复杂性类在文献中也常提到。

内容提要

非一致的多项式时间（P/poly）是指由只能处理固定长度输入的机器所执行的高效计算。基本定义中忽略了构造这类设备的复杂性（即一致性条件）。在更精确的定义中，对所谓“接受建议的机器”的非一致性程度进行了量化。

多项式时间层级（PH）是对 NP 类的推广，它考虑的是这样一种量化布尔公式，其中交替使用固定数量的存在量词和全称量词。普遍认为这些量词的交替使用可以增加这类布尔公式的表现力。

与这两种类型都相关的一个有趣结论是，如果 NP 包含在 P/poly 中，则多项式时间层级会衰减到其第二层。这个结论通常被理解为支持非一致性与 P-vs-NP 问题无关这样一种普遍认识，即虽然 P/poly 类超出了 P 类的范畴，但它仍不能完全包含 NP。

3.2.3 节中将指出这个结论的一种例外，此时，P/poly（见 3.1 节）与多项式时间层级（见 3.2 节）相互独立。

3.1 非一致的多项式时间

本节考虑非一致的多项式时间的两种不同的形式化定义，它们分别基于 1.2.4 节中描述的两种非一致的模型。在讨论多项式界限的复杂度时，将具体指定使用何种非一致的模型，这些设备在 1.2.4 节已给出。最后要指出，可能存在一类计算问题，同时符合两种（多项式界）形式化定义，此时，这类问题是多项式时间算法可解问题的一个严格意义上的超集。

两种非一致的模型是布尔电路和“接受建议的机器”（分别见于 1.2.4.1 节和 1.2.4.2

节)。我们仍重点讨论将两种模型限定于多项式复杂度的情形,并考虑(非一致的)多项式规模电路和接受(非一致的)多项式界长度建议的多项式时间算法。

讨论非一致多项式规模电路的主要动机是,对其在计算上的限制蕴含了类似的对多项式时间算法的限制。预期目标是,如同在数学和自然科学中常见的情况,去掉不太重要且尚未被充分理解的辅助条件(如一致性),^①可能会得到极其丰富的结论。特别地,(非一致的)电路模型便于从较低层次分析计算的演化,并允许使用组合技术。这种方法的优点将在研究带约束条件的电路类型时说明(见附录 B.2.2 和附录 B.2.3)。

研究接受多项式界长度建议的多项式时间算法的主要动机是:这类算法有助于为辅助信息建立模型,而一些有用的高效策略将使用这些信息。我们指出两个这样的环境。在密码学(见附录 C)中,用这种算法为敌手可以利用的辅助信息建模。在去随机化中(见 8.3 节),用这种算法为随机算法的输入建模。另外,接受建议的多项式算法模型便于对从零到多项式的非一致性程度进行定量研究。

3.1.1 布尔电路

建议读者参考 1.2.4.1 节中对布尔电路(类)以及可由布尔电路计算的函数的定义。为了更加具体和简洁,本节假设所有电路都是有界扇入的。首先重述 1.2.4.1 节中的结论。

定理 3.1 (电路求值) 存在一个多项式时间算法,给定电路 $C: \{0,1\}^n \rightarrow \{0,1\}^m$ 及 n 比特串 x , 算法返回 $C(x)$ 。

算法体现了电路对于给定输入的计算,它通过“确定赋值”的过程来发挥作用。该过程为电路中每个结点赋值,赋值的依据是其孩子结点(或者对于输入端顶点,依据输入中的相关比特来赋值)。

用电路规模作为复杂性量度

回顾在 1.2.4.1 节给出的电路复杂性定义中,电路的规模是指其中边的个数,而电路描述的长度基本上与边的数量呈线性关系,即规模为 s 的电路通常用其中边和顶点的列表来表示,因此,电路描述的长度为 $O(s \log s)$ 。我们感兴趣的是能解计算问题的电路类,称电路类 $(C_n)_{n \in \mathbb{N}}$ 能计算函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$, 如果对任意 $x \in \{0,1\}^*$, 有 $C_{|x|}(x) = f(x)$ 。电路类的规模复杂度(Size Complexity)定义为函数 $s: \mathbb{N} \rightarrow \mathbb{N}$, 其中 $s(n)$ 是 C_n 的规模。函数 f 的电路复杂度(Circuit Complexity), 记作 s_f , 是指能计算 f 的最小规模电路类的规模复杂度。以下给出严格定义。

定义 3.2 (电路复杂度) 函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 的电路复杂度定义为函数 $s_f: \mathbb{N} \rightarrow \mathbb{N}$, 其中 $s_f(n)$ 表示将 f 限定为 n 比特输入串时,能计算 f 的最小电路。

这个定义并未明确体现非一致性,因为其中对于计算函数的不同长度输入的各种电路间的关系没有提出任何限制条件。

定义 3.2 的一个有趣特征是,与一致性计算模型不同,这里考虑的是函数的实际复杂性,而非其复杂性上限(见 1.2.3.4 节及 4.2.1 节)。这是由于电路模型中没有“自由参数”(如在

^① 普遍认为非一致性问题与 P-vs-NP 问题无关,即通过证明 $P \neq \mathcal{NP}$ 来解决后者并不比证明 NP 类中不含多项式规模的电路更容易。进一步的讨论见附录 B.2 和 3.2.3 节。

一致性模型中使用的算法所需的各种参数)。^①

我们感兴趣的是可以由多项式规模电路求解的问题类。一个问题可被多项式规模的电路求解, 如果它可由具有多项式电路复杂度的函数 f 求解的话 (即存在多项式 p , 使得对任意 $n \in \mathbb{N}$, 有 $s_f(n) \leq p(n)$)。

题外话: 一致性类

多项式规模电路类 $(C_n)_{n \in \mathbb{N}}$ 称为一致的, 如果给定 n , 可以在时间 $\text{poly}(n)$ 内计算电路 C_n 。

更一般地, 有

定义 3.3 (一致性) 电路类 $(C_n)_{n \in \mathbb{N}}$ 称为一致的, 如果存在一个算法, 对于输入 n 能输出 C_n , 且算法的计算步数是 C_n 规模的多项式。

注意: 人们可能会考虑更强一致性的概念, 例如, 要求存在多项式时间算法, 对于输入的 n 和 v , 输出顶点 v 的标号及其孩子结点列表 (或指出 v 不是 C_n 中的顶点)。进一步的讨论见 5.2.3 节。回到定义 3.3, 注意到事实上一致性电路类的计算过程可以用一个一致的计算设备来模仿。

命题 3.4 如果一个问题可由一致的多项式规模电路求解, 则它也可由多项式时间算法求解。

1.2.4.1 节中暗示了, 上述命题的逆命题也成立。由定理 2.21 (及定理 3.6) 的证明过程易得出该结论。

证明: 算法对于输入 x 的计算分为两步。第一步, 对于输入 $n = \overset{\text{def}}{|x|}$, 调用由一致性条件所保证的算法, 并得到电路 C_n ; 第二步, 对于输入 C_n 和 x 调用电路求值算法 (定理 3.1), 得到 $C_n(x)$ 。由于 C_n 的规模 (及其描述) 的长度是 n 的多项式, 故算法的每一步都在多项式时间内运行 (即关于 $n = |x|$ 的多项式)。因此, 上述算法可以模仿 $C_{|x|}(x)$ 的计算过程, 并在关于其输入 (即 x) 长度的多项式时间内完成。 ■

3.1.2 接受建议的机器

对于计算设备的“非一致程度”而言, 一般类型的 (可能是非一致的) 多项式规模电路类及一致的多项式规模电路类是两个极端。直观上, 对于前者, 非一致性上限只取决于设备的规模, 而对于后者, 非一致性为 0。这里考虑一种计算模型, 能将计算设备的规模与其非一致程度分离, 非一致程度的取值范围从 0 到设备规模。具体地, 我们考虑“接受非一致性建议”的算法, 它只依赖于输入长度。此时的非一致程度定义为相应建议的长度 (是输入长度的一个函数)。因此, 我们将定义 1.12 限定到多项式时间算法。

定义 3.5 (非一致的多项式时间和 P/poly) 称函数 f 在多项式时间内利用长度为 $l: \mathbb{N} \rightarrow \mathbb{N}$ 的建议可解, 如果存在一个多项式时间算法 A 和一个无限长的建议序列 $(a_n)_{n \in \mathbb{N}}$, 满足以下条件。

- (1) 对任意 $x \in \{0, 1\}^*$, 有 $A(a_{|x|}, x) = f(x)$ 。
- (2) 对任意 $n \in \mathbb{N}$, 有 $|a_n| = l(n)$ 。

^① 一致性模型中的“自由参数”包括算法描述的长度及其字母表的大小。注意: 这些“自由参数”支持如习题 4.4 中的线性加速结论, 这些结论不规定函数 (一致性) 的精确复杂度。

称一个计算问题在多项式时间内利用长为 l 的建议可解, 如果求解该问题的算法可在多项式时间内运行。用符号 P/l 表示在多项式时间内利用长为 l 的建议可解的判定问题类, 而用 $P/poly$ 表示所有 P/p 的并集, p 为任意多项式。

显然, $P/0 = P$, 但是当允许某种(非空)建议时类中的元素会增加(见定理 3.7), 而当建议长度与时间复杂度相当时, 能得到一个等价于电路复杂度的公式(见定理 3.6)。我们强调接受建议机器的定义有很强的弹性, 它允许独立地规定时间复杂性和建议长度(事实上, 这样会引入更复杂的形式化定义, 因此, 当需要考虑两种复杂性量度均为多项式情形时, 更倾向于使用电路的定义)。

多项式规模电路类之间的关系

前面已经有所暗示, 由多项式时间算法利用多项式界建议可解的问题类等价于用多项式规模电路可解的问题类。具体而言, 关于判定问题有如下结论。

定理 3.6 判定问题属于 $P/poly$ 当且仅当其可由多项式规模电路类求解。

更一般地, 对任意函数 t , 以下证明过程在接受 t 长度建议的多项式时间机器与规模为 t 的多项式的电路类之间建立起等价关系。

证明概要: 假设一个问题由多项式时间算法 A 利用多项式界的建议序列 $(a_n)_{n \in \mathbb{N}}$ 可解。利用定理 2.21 证明中所采用的方法, 可以得到解同一问题的多项式规模电路类。具体而言, 观察到对于 $A(a_{|x|}, x)$ 的计算可以用规模为 $\text{poly}(|x|)$ 的电路来模仿, 只需将 x 作为输入, 并引入 $a_{|x|}$ 即可。也就是说, 可以构造一个电路 C_n , 使得对任意 $x \in \{0, 1\}^n$, 有 $C_n(x) = A(a_n, x)$ (类似于定理 2.21 证明中构造 C_x 的方法, 其中对任意足够长的 y , 有 $C_x(y) = M_R(x, y)$)。^①

另一方面, 给定一个多项式规模电路类, 可以将相关电路的描述当作建议来构造多项式时间接受建议的机器, 用以模仿这个类。具体地, 将定理 3.1 中的模仿算法修改成为接受建议的机器, 当给定建议 α 和输入 x 时, 将 α 当作电路 C 的描述, 并计算 $C(x)$ 。换言之, 我们利用这样一个事实, 即电路规模 s 可用长为 $O(s \log s)$ 的串来描述, 其中对数因子是由于含 v 个顶点和 e 条边的图可用长为 $2e \log_2 v$ 的串来描述。□

另一种角度

称集合 S 是稀疏的, 如果存在多项式 p , 使得对任意 n 有 $|S \cap \{0, 1\}^n| \leq p(n)$ 。注意到 $P/poly$ 等价于可 Cook-归约为稀疏集的复杂性类(见习题 3.2)。因此, SAT 可以 Cook-归约为一个稀疏集当且仅当 $\mathcal{NP} \subset P/poly$ 。相反, SAT 可以 Karp-归约为稀疏集当且仅当 $\mathcal{NP} = P$ (见习题 3.12)。

$P/poly$ 的功能

作为定理 1.13 (其中主要讨论建议而忽略了接受建议机器的时间复杂度)的后续, 我们证明以下这个(更强的)结论。

定理 3.7 (建议的功能, 重述) 类 $P/l \subseteq P/poly$ 中包含了 P 以及一些不可判定的问题。

实际上, $P/l \subset P/poly$, 另外, 利用计数, 可以证明对任意两个多项式界函数 $l_1, l_2: \mathbb{N} \rightarrow \mathbb{N}$, 且 $l_2 - l_1 > 0$ 无上界, 则 P/l_1 是 P/l_2 的真子集, 见习题 3.3。

^① 注意: a_n 是电路 C_n 中唯一“非一致”的部分。因此, 如果算法 A 不接受任何建议(即对任意 n , 有 $a_n = \lambda$), 就得到了一致性的电路类。

证明: 显然, $P = P/0 \subseteq P/1 \subseteq P/\text{poly}$ 。为证明 $P/1$ 中还包含一些不可判定问题, 参考定理 1.13 的证明, 其中证明了不可计算的布尔函数的存在性只取决于其输入长度。这就是说, 存在一个不可判定集合 $S \subset \{0,1\}^*$, 使得对任意一个由相同长度串构成的对 (x,y) , $x \in S$ 当且仅当 $y \in S$ 。换言之, 对任意 $x \in \{0,1\}^*$, 有 $x \in S$ 当且仅当 $1^{|x|} \in S$ 。但是这样一个集合的成员判定可由接受 1 比特建议的机器在多项式时间内完成, 也就是说, 考虑满足 $A(a,x) = a$ 的算法 A (对 $a \in \{0,1\}$, $x \in \{0,1\}^*$) 和建议序列 $(a_n)_{n \in \mathbb{N}}$, 使得 $a_n = 1$ 当且仅当 $1^n \in S$ 。注意到, 事实上, $A(a_{|x|}, x) = 1$ 当且仅当 $x \in S$ 。 ■

3.2 多项式时间层级

多项式时间层级是对 $\mathcal{NP}$ 类一种很自然的推广。有趣的是, 当且仅当 $\mathcal{NP} = P$ 时, 这种推广可以退化为 P , 进一步地, 这是已知对于 $\mathcal{NP}$ 类具有这种性质的最大推广。我们从一个非正式的讨论开始, 在 3.2.1 节再引入正式定义。

$\mathcal{NP}$ 中的集合可以看作是正确断言的集合, 这些断言的量化布尔公式中只包含存在量词。也就是说, 称集合 S 属于 $\mathcal{NP}$, 如果存在一个 Karp-归约, 将 S 映射为判定一个存在性量化布尔公式是否正确 (即该归约将实例 x 映射为形如 $\exists y_1 \cdots \exists y_m \Phi_x(y_1, y_2, \dots, y_{m(x)})$ 的公式。)

$\mathcal{NP}$ 问题难解的假设似乎是存在量词序列太长。当然, 如果还有人 (“证明者”) 想要利用足够的赋值 (对 y_i) 来证明 (只要这种赋值存在) 这一点, 则情况要好得多。因为此时可以高效地验证存在性量化布尔公式的正确性。

但是如果我们要验证全称量化布尔公式 (即形如 $\forall y_1 \forall y_2 \cdots \forall y_m \Phi(y_1, y_2, \dots, y_m)$ 的公式) 的正确性时又该如何做呢? 此时, 似乎要使用一个完全不同的实体: 需要一个 “反驳者” 来提供驳斥 (无论是否存在), 并且要相信如果没有提供驳斥, 则一定是不存在这样的驳斥 (从而, 该全称量化公式是正确的)。事实上, 这种新的环境 (“驳斥系统”) 与证明系统有着本质区别: 在证明系统中, 只能通过 (对断言的) 证明来确认某个断言的正确性, 而在 “驳斥系统” 中, 我们相信 “反驳者” 提供了断言错误的证据。^① 进一步地, 似乎没有办法将一种环境 (即证明系统) 转化为另一种 (驳斥系统)。

我们会进一步地考虑一个更复杂的系统, 其中使用两个代理: 一个是 “支持者”, 要提供某个断言成立的证据; 另一个是 “反对者”, 要证明该断言不成立。这两个代理进行辩论 (或争论), 通过信息交互使我们 (裁判) 做出裁决, 相信其说法。在这个系统中, 被证明的断言具有一般量化公式的形式, 但是交替使用不同的量词, 其中量词转换的次数等于在上述过程中交互的圈数。我们强调每个同一类型量词序列的确切长度并不重要, 重要的是转换的次数, 记作 k 。

上述的交替系统可以看作是一个两方游戏, 人们会问哪一方在 k 次交互后能取胜。一般来说, 考虑一个 (0-1, zero-sum) 两方游戏, 游戏的位置可以高效更新 (通过给定的动作), 并高效赋值。对于这样一个游戏, 给定初始位置, 要问是否第一方具有 (k -动作) 取胜策略。

① 更严格地说, 在证明系统中, 合理性条件只取决于验证者的行为, 而完备性还取决于证明者的行为 (即采用了充分的策略)。相反, 在 “拒斥系统” 中, 合理性条件取决于驳斥者的行为, 而完备性则与驳斥者的行为无关。

对于某个给定的 k 能回答这个问题似乎并不一定意味着能对 $k+1$ 也做出回答。下面形式化地描述上述讨论。

3.2.1 量词的转换

在以下定义中, 将前面提到的命题公式 Φ_x 用输入 x 来代替 (相应地, Karp-归约和公式赋值算法也由验证算法 V 来代替 (见习题 3.7))。这样做是为了使与 $\mathcal{NP}$ 的定义进行比较时更加透明 (同时也为了适应标准表述)。我们还会将同一类型的布尔量词序列用相应的量词来代替, 这些量词能对所有具相应长度的串进行量化。

定义 3.8 (Σ_k 类) 对于给定正整数 k 和判定问题 S , 如果存在多项式 p 及多项式时间算法 V , 使得 $x \in S$ 当且仅当

$$\exists y_1 \in \{0,1\}^{p(|x|)} \forall y_2 \in \{0,1\}^{p(|x|)} \exists y_3 \in \{0,1\}^{p(|x|)} \dots Q_k y_k \in \{0,1\}^{p(|x|)}$$

满足

$$V(x, y_1, \dots, y_k) = 1$$

则称判定问题 S 属于 Σ_k 类。其中 k 为奇数时, Q_k 是存在量词, 否则, Q_k 为全称量词。

注意: $\Sigma_1 = \mathcal{NP}$ 且 $\Sigma_0 = \mathcal{P}$ 。多项式时间层级 (Polynomial-time Hierarchy), 记作 $\mathcal{PH}$, 定义为所有上述类的并集 (即 $\mathcal{PH} = \bigcup_k \Sigma_k$), 其中 Σ_k 通常指 $\mathcal{PH}$ 的第 k 层。多项式时间层级中的层可以递归地定义, 即基于 $\Pi_k \stackrel{\text{def}}{=} \text{co}\Sigma_k$ 来定义 Σ_{k+1} , 其中 $\text{co}\Sigma_k \stackrel{\text{def}}{=} \{ \{0,1\}^* \setminus S : S \in \Sigma_k \}$ (见式 (2.4))。

命题 3.9 对任意 $k \geq 0$, 集合 S 属于 Σ_{k+1} 当且仅当存在多项式 p 和集合 $S' \in \Pi_k$, 使得 $S = \{x : \exists y \in \{0,1\}^{p(|x|)}, \text{ 满足 } (x, y) \in S'\}$ 。

证明: 假设 S 属于 Σ_{k+1} , 令 p 和 V 如定义 3.8 所述, 定义 S' 由满足 $|y| = p(|x|)$ 且

$$\forall z_1 \in \{0,1\}^{p'(|x|)} \exists z_2 \in \{0,1\}^{p'(|x|)} \dots Q_k z_k \in \{0,1\}^{p'(|x|)}$$

使得

$$V(x, y, z_1, \dots, z_k) = 1$$

的对 (x, y) 组成。注意: $x \in S$ 当且仅当存在 $y \in \{0,1\}^{p(|x|)}$ 使得 $(x, y) \in S'$, 且 $S' \in \Pi_k$ (见习题 3.6)。

另一方面, 假设对于多项式 p 和集合 $S' \in \Pi_k$, 有 $S = \{x : \exists y \in \{0,1\}^{p(|x|)}, \text{ 使得 } (x, y) \in S'\}$, 则对于 p' 和 V' , 有 $(x, y) \in S'$ 当且仅当 $|y| = p(|x|)$ 且

$$\forall z_1 \in \{0,1\}^{p'(|x|)} \exists z_2 \in \{0,1\}^{p'(|x|)} \dots Q_k z_k \in \{0,1\}^{p'(|x|)}$$

使得

$$V'(x, y, z_1, \dots, z_k) \neq 1$$

(又见习题 3.6)。通过适当地编码 (将 y 及 z_i 映射到长为 $\max(p(|x|), p'(|x|))$ 的串), 并对 V' 做简单地修改, 可以得到 $S \in \Sigma_{k+1}$ 。

在 k -动作游戏中确定胜者

定义 3.8 可以理解为在某个高效的两方游戏中确定胜者的复杂性。具体地, 这里所说的

两方游戏要满足以下 3 个条件。

(1) 各参与方交替行动影响游戏的位置, 对每次行动描述的长度是当前位置描述长度的多项式。

(2) 基于前一个位置及参与方的当前行为, 在多项式时间内更新当前位置。^①

(3) 可以在多项式时间内确定每个位置中的胜者。

注意: 能使第一方在 k -动作游戏中获胜的初始位置集合属于 Σ_k 。具体地, 将该集合记作 G , 初始位置 x 在 G 中的条件是: 存在第一方的动作 y_1 , 使得对第二方的每个回应动作 y_2 , 存在第一方动作 y_3 , 以此类推, 使得在 k 次动作后, 各方到达的位置能使第一方获胜, 其中最终位置是根据上述的条件 (2) 来确定的, 而获胜方由条件 (3) 确定。^② 因此, $G \in \Sigma_k$ 。另一方面, 注意到任意集合 $S \in \Sigma_k$ 可看作是初始位置集合 (在适当的游戏中), 其中第一方在 k 次动作后获胜。具体来说, $x \in S$ 的条件是: 从初始位置 x 开始, 对第一方而言存在一个动作 y_1 , 使得对任意相应的第二方动作 y_2 , 存在第一方动作 y_3 , 以此类推, 使得在 k 次动作后, 到达第一方获胜的位置, 而这里的最终位置定义为 $(x, y_1, \dots, y_k)$, 而获胜者则由谓词 V 确定 (如定义 3.8 所述)。

PH 及 P-vs-NP 问题

我们强调一个事实, $\mathcal{PH} = \mathcal{P}$ 成立当且仅当 $\mathcal{P} = \mathcal{NP}$ 。事实上, $\mathcal{PH} = \mathcal{P}$ 蕴含着 $\mathcal{P} = \mathcal{NP}$ 纯粹是从句法角度考虑的, 其逆命题可由命题 3.9 得出 (还可见于命题 3.10 证明中的第二部分)。^③ $\mathcal{P} = \mathcal{NP}$ 蕴含着 $\mathcal{PH} = \mathcal{P}$ 表明可以通过证明 $\mathcal{PH} \neq \mathcal{P}$ 来证明 $\mathcal{P} \neq \mathcal{NP}$ 。因此, 为了区分两个类 (即 $\mathcal{P} \neq \mathcal{NP}$), 可以先区分其中较小的类与 (即 $\mathcal{P}$) 与另一类 (即 $\mathcal{NP}$) 的超集 (即 $\mathcal{PH}$)。

其他等式的退化效应

对 $\mathcal{NP} \neq \text{coNP}$ 的假设进行扩展, 通常对任意 $k \in \mathbb{N}$ 猜想 $\Sigma_k \neq \Pi_k$ 。如果这个猜想不成立, 则将引起多项式时间层级向相应层级的退化。

命题 3.10 对任意 $k \geq 1$, 如果 $\Sigma_k = \Pi_k$, 则 $\Sigma_{k+1} = \Sigma_k$, 从而有 $\mathcal{PH} = \Sigma_k$ 。

上述命题的逆命题也成立 (即 $\mathcal{PH} = \Sigma_k$ 蕴含着 $\Sigma_{k+1} = \Sigma_k$ 及 $\Sigma_k = \Pi_k$)。当然, $k=0$ 时, 命题 3.10 的第一部分不成立, 但是第二部分对 $k=0$ 也成立 (即 $\Sigma_1 = \Sigma_0$ 蕴含着 $\mathcal{PH} = \Sigma_0$)。

证明: 设 $\Sigma_k = \Pi_k$, 首先证明 $\Sigma_{k+1} = \Sigma_k$ 。对 Σ_{k+1} 中的任意集合 S , 由命题 3.9, 存在多项式 p 和集合 $S' \in \Pi_k$, 使得 $S = \{x: \exists y \in \{0,1\}^{p(|x|)}, \text{使得 } (x,y) \in S'\}$ 。利用这个假设, 可以得到 $S' \in \Sigma_k$, 所以 (由命题 3.9 及 $k \geq 1$) 存在多项式 p' 和集合 $S'' \in \Pi_{k-1}$, 使得

① 注意: 由于考虑的是常数动作, 所有最终位置的长度上界是初始位置长度的多项式, 因此在将复杂性看作初始位置长度的函数时, 所有的项具有等价形式。后一种形式允许向多项式动作次数的游戏的平滑推广 (见 5.4 节), 其中有必要将所有复杂性用初始位置长度表示。

② 设 U 为条件 (2) 中的更新算法, W 为条件 (3) 中确定胜者的算法, 则最终位置可通过对 $i=1,2,\dots,k$ 计算 $x_i \leftarrow U(x_{i-1}, y_i)$ 而得到 (其中 $x_0 = x$), 其中获胜者为 $W(x_k)$ 。注意: 在条件 (1) 中, 存在多项式 p 使得对任意 $i \in [k]$ 有 $|y_i| \leq p(|x_i|)$, 从而有 $|y_i| \leq \text{poly}(|x|)$ 。利用适当的编码, 可以得到多项式时间算法 V , 使得 $V(x, y_1, \dots, y_k) = W(x_k)$, 其中 $x_k = U(\dots U(U(x, y_1), y_2), y_3), \dots, y_k)$ 。

③ 我们强调这种蕴含关系并非由从 $\mathcal{PH}$ 到 $\mathcal{NP}$ 的 Cook-归约得出, 事实上, 这样的 Cook-归约只存在于 $\mathcal{PH}$ 的一个子类 (是 $\Sigma_2 \cap \Pi_2$ 的子集) 中。

$$S' = \{x' : \exists y' \in \{0,1\}^{p'(|(x,y)|)}, \text{ 使得 } (x', y') \in S''\}$$

从而有

$$S = \{x : \exists y \in \{0,1\}^{p(|x|)} \exists z \in \{0,1\}^{p'(|(x,y)|)}, \text{ 使得 } ((x,y), z) \in S''\}$$

通过消去两个相邻的存在量词（并再次利用命题 3.9），可以得到 $S \in \Sigma_k$ 。这就证明了命题的第一部分。

至于第二部分，注意：已知 $\Sigma_{k+1} = \Sigma_k$ （或等价地， $\Pi_{k+1} = \Pi_k$ ），从而有 $\Sigma_{k+2} = \Sigma_{k+1}$ （再次利用命题 3.9），类似地，对任意 $j \geq k$ 有 $\Sigma_{j+2} = \Sigma_{j+1}$ ，所以 $\Sigma_{k+1} = \Sigma_k$ 蕴含着 $\mathcal{PH} = \Sigma_k$ 。■

可以 Cook-归约为 NP 的判定问题

多项式时间层级包含了所有可以 Cook-归约到 NP 的判定问题（见习题 3.4）。下面将证明，许多自然问题属于这种类型。回顾在 2.2.2 节中定义了两种类型的优化问题，并证明了在某些自然条件下，这两种类型在计算上等价（利用 Cook-归约）。具体而言，一种类型是寻找超过某个给定阈值的解，另一种是寻找最优解。2.3 节提出了几个属于第一种类型的问题，并证明了其 NP-完全性。注意：这并不意味着第二种类型中的相应版本也属于 NP。例如，我们讨论了对于给定图 G ，判定其中是否包含规模为给定整数 K 的团，并证明了这个问题的 NP-完全性。相反，判定 K 是否为图 G 中团的最大规模这一问题是否属于 NP 尚且未知（貌似不可能），虽然它可以 Cook-归约为 NP。因此，可以 Cook-归约到 NP 的判定问题类中可能包含许多不属于 NP 的自然问题。多项式时间层级则包含了所有这类问题。

完全问题及其与 AC0 的关系

注意：当量词转换次数有上界时，一个量化布尔公式能为各种多项式时间层级提供完全问题（见习题 3.7）。我们还指出，这些公式与强非一致的常数深度电路间存在对应关系，这些电路具有无限扇入，且输入为对应（非量化）公式的真值表（见习题 3.8）。

3.2.2 非确定型预言机

多项式时间层级通常用非确定多项式时间（预言）机的术语来定义，该预言机能访问同一层级中较低层的集合。这种机器的定义结合了非确定（多项式时间）机器（见定义 2.7）与预言机（见定义 1.11）的概念。具体地，对于预言 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ ，非确定型预言机 M 以及串 x ，考虑当 M 能访问预言 f 时，是否存在对于输入 x 的可接受的（非确定型）计算。由非确定型多项式时间（预言）机在访问 f 时能接受的集合定义为 $\mathcal{NP}^f$ （注意：这个符号是有意义的，因为可以将 NP 类与一组能扩展为预言机的机器相关联）。对任意的判定问题类 C ，用 $\mathcal{NP}^C$ 表示 $\mathcal{NP}^f$ 之并集能处理的 C 中所有判定问题 f 。以下结论给出多项式时间层级的另一种定义。

命题 3.11 对任意 $k \geq 1$ ，有 $\Sigma_{k+1} = \mathcal{NP} \Sigma_k$ 。

显然， $\Sigma_1 = \mathcal{NP} = \Sigma_0$ ，但这是一种平凡情形（即 $\Sigma_1 = \mathcal{NP} = \mathcal{NP}^P = \mathcal{NP}^{\Sigma_0}$ ）。

证明：前半部分（即 $\Sigma_{k+1} \subseteq \mathcal{NP} \Sigma_k$ ）可以直接证明：对任意 $S \in \Sigma_{k+1}$ ，令 $S' \in \Pi_k$ 及 p 如命题 3.9 中所述，即 $S = \{x : \exists y \in \{0,1\}^{p(|x|)}, \text{ 使得 } (x,y) \in S'\}$ ，考虑一个非确定预言机，对于输入 x ，非确定地生成 $y \in \{0,1\}^{p(|x|)}$ ，并当且仅当 $(x,y) \in S'$ 时接受输入 (x,y) （如预言所指示

的)。该机器的存在说明了 $S \in \mathcal{NP}^{\Pi_k} = \mathcal{NP}^{\Sigma_k}$ ，其中可以通过让预言机在由预言给出的每个（二元）的回答间变换来使等式成立。^①

至于后半部分，即 $\mathcal{NP}^{\Sigma_k} \subseteq \sum_{k+1}$ ，可以对定理 2.35 的证明思想进行推广（其中针对的是 $\mathcal{P}^{\mathcal{NP}} \cap \text{co}\mathcal{NP}$ ）。具体地，考虑任意集合 $S \in \mathcal{NP}^{\Sigma_k}$ ，令 M 为非确定的多项式时间预言机，当给定预言访问 $S' \in \sum_k$ 时接受 S 。注意：（与前面构造的机器不同）机器 M 可能会向 S' 发出几个询问，而这些询问可能是基于对以前预言应答而确定的。^②为简单起见，不失一般性，假设机器 M 在开始执行时，会（非确定地）猜测所有的预言应答，并只接受与其猜测相匹配的应答。因此， M 向对预言的询问由其输入（记作 x ）和其非确定的选择（记作 y ）确定。将 M 发出的第 i 次询问记作 $q^{(i)}(x, y)$ （对于输入 x 和非确定选择的 y ），并将相应的对应答的猜测（是 y 中一个比特）记作 $a^{(i)}(x, y)$ 。因此， M 接受 x 当且仅当存在 $y \in \{0, 1\}^{\text{poly}(|x|)}$ 满足以下两个条件。

(1) 机器 M 对于输入 x 和非确定选择的 y ，当第 i 个询问的应答为 $a^{(i)}(x, y)$ 时接受 x 。将相应的（“接受”）断言记作 $A(x, y)$ ，它是多项式时间可计算的。

我们强调在这里并没有假设这些应答 $a^{(i)}(x, y)$ 与 S' 给出的应答一致，这是下一个条件要考虑的。本条件只针对 M 对某个给定输入的判定，这构成一个非确定的选择序列，并给出了特定的预言应答。

(2) $a^{(i)}(x, y)$ 中每个比特都与 S' 一致，即对任意 i ， $a^{(i)}(x, y) = 1$ 当且仅当 $q^{(i)}(x, y) \in S'$ 。

将 M 发出的询问数量（对输入 x 及非确定选择的 y ）记作 $q(x, y) \leq \text{poly}(|x|)$ ，则有 $x \in S$ 当且仅当

$$\exists y \left(A(x, y) \wedge \bigwedge_{i=1}^{q(x, y)} \left((a^{(i)}(x, y) = 1) \Leftrightarrow (q^{(i)}(x, y) \in S') \right) \right) \quad (3.1)$$

将 S' 的验证算法记作 V' ，则式 (3.1) 等价于

$$\exists y \left(A(x, y) \wedge \bigwedge_{i=1}^{q(x, y)} \left((a^{(i)}(x, y) = 1) \Leftrightarrow \exists y_1^{(i)} \forall y_2^{(i)} \cdots Q_k y_k^{(i)} V'(q^{(i)}(x, y), y_1^{(i)}, \dots, y_k^{(i)}) = 1 \right) \right)$$

可以重新排列上述表达式，使其符合 Σ_{k+1} 的定义，从而完成证明。具体细节如下。

首先用 $(E_1 \wedge E_2) \vee (\neg E_1 \wedge \neg E_2)$ 代替其中的每个子表达式 $E_1 \Leftrightarrow E_2$ ，再将量词取出^③。用这种方法可以得到一个含 $k+1$ 次量词转换的表达式，它从存在量词开始（注意：我们得到的是含 $k+1$ 次量词转换的表达式，而非 k 次，这是因为 $\neg a^{(i)}(x, y) = 1$ 的情况引入了形如 $\neg \exists y_1^{(i)} \forall y_2^{(i)} \cdots Q_k y_k^{(i)} V'(q^{(i)}(x, y), y_1^{(i)}, \dots, y_k^{(i)}) = 1$ 的表达式，而这又等价于 $\forall y_1^{(i)} \exists y_2^{(i)} \cdots \overline{Q_k} y_k^{(i)} \neg V'(q^{(i)}(x, y), y_1^{(i)}, \dots, y_k^{(i)}) = 1$ ）。然后，再具体计算所有 $q^{(i)}(x, y)$ （以及 $a^{(i)}(x, y)$ ）的值，并在新

① 不要被预言类对于求补可能不封闭这一事实所迷惑。从预言机角度看，预言仅仅是一个函数，而机器可能对其应答做出任何处理（特别是可能对其求逆）。

② 这不同于证明 $\Sigma_{k+1} \subseteq \mathcal{NP}^{\Sigma_k}$ 时所使用的机器。

③ 例如，对于谓词 p_1 和 p_2 ，表达式 $\exists y(p_1(y) \Leftrightarrow \exists z p_2(y, z))$ 等价于表达式 $\exists y((p_1(y) \wedge \exists z p_2(y, z)) \vee (\neg p_1(y) \wedge \neg \exists z p_2(y, z)))$ ，后者又等价于 $\exists y \exists z' V z''((p_1(y) \wedge p_2(y, z')) \vee (\neg p_1(y) \wedge \neg p_2(y, z'')))$ 注意将 $\bigwedge_{i=1}^l \exists y^{(i)} \forall z^{(i)} p(y^{(i)}, z^{(i)})$ 中的量词外置，得到的表达式形如 $\exists y^{(1)}, \dots, y^{(l)} \forall z^{(1)}, \dots, z^{(l)} \bigwedge_{i=1}^l P(y^{(i)}, z^{(i)})$ 。

的验证算法 V 中调用 V' (及其他逻辑运算) 多项式次, 从而证明了 $S \in \sum_{k+1}$ 。

一个普遍问题—— $C_1^{C_2}$ 的含义是什么?

从上述讨论中很清楚地看到, $C_1^{C_2}$ 类可以由两个复杂性类 C_1 和 C_2 来定义, 并且 C_1 与一类机器相关, 这类机器可以很自然地推广为预言机。事实上, $C_1^{C_2}$ 类的定义并非基于 C_1 , 而是类似于 C_1 的类。具体地, 设 C_1 为可以由某种有资源上限 (如时间及/或空间上限) 的 (确定或非确定的) 机器识别 (或接受) 的集合。我们考虑类似的预言机 (与该机器类型相同, 资源上限也相同), 如果存在足够的预言机 M_1 (即具有相同类型和资源上限) 以及集合 $S_2 \in C_2$ 使得 $M_1^{S_2}$ 接受集合 S , 则称 $S \in C_1^{C_2}$ 。

可以 Cook-归约到 NP 的判定问题, 重述

利用上述符号, 将可以 Cook-归约到 $\mathcal{NP}$ 的判定问题记作 $\mathcal{P}^{\mathcal{NP}}$, 并且它是 $\mathcal{NP}^{\mathcal{NP}} = \Sigma_2$ 的子集 (见习题 3.9)。相反, 可以 Karp-归约到 $\mathcal{NP}$ 的判定问题类其实等于 $\mathcal{NP}$ 。

全局观点

利用上述记号, 并根据习题 3.9, 可以得到对任意 $k \geq 1$, 有 $\Sigma_k \cup \Pi_k \subseteq \mathcal{P}^{\Sigma_k} \subseteq \Sigma_{k+1} \cap \Pi_{k+1}$ 。如图 3.1 所示, 其中描述了这一结论, 并假设所有的包含关系都是真子集。

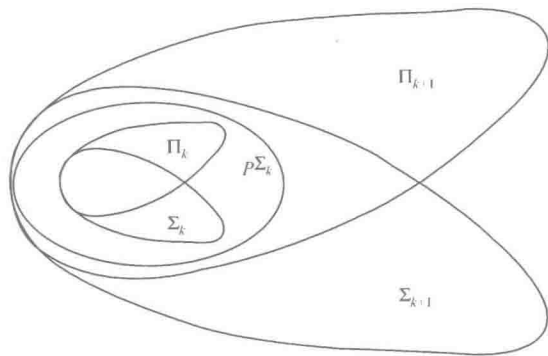


图 3.1 多项式时间层级中的两层

3.2.3 P/poly-vs-NP 问题及 PH 类

如 3.1 节所述, 定义 $\mathcal{P}/\text{poly}$ 的主要动机是可以利用它将 $\mathcal{P}$ 类从 $\mathcal{NP}$ 中区分出来 (通过证明 $\mathcal{NP}$ 不属于 $\mathcal{P}/\text{poly}$, 而它却是 $\mathcal{P}$ 的 (严格) 超集)。当 $\mathcal{P}/\text{poly}$ 的范围远远超出 $\mathcal{P}$ 时 (特别是其中包含不可判定问题时), 人们可能会怀疑这种方法是否会提问过多 (因为即使 $\mathcal{P} \neq \mathcal{NP}$, 也有可能 $\mathcal{NP}$ 属于 $\mathcal{P}/\text{poly}$)。通常认为附加的非一致性与 P-vs-NP 问题无关。在理想情况下, 我们希望仅当 $\mathcal{P} = \mathcal{NP}$ 时才有 $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, 这也可以说成多项式时间层级会退化到第零级。以下结论似乎接近于这种暗示, 其中表明仅当多项式时间层级退化到第二级时才有 $\mathcal{NP} \subset \mathcal{P}/\text{poly}$ 。

定理 3.12 如果 $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, 则 $\Sigma_2 = \Pi_2$ 。

$\Sigma_2 = \Pi_2$ 实际上蕴含了 $\text{PH} = \Sigma_2$ (见命题 3.10), 因此, 非一致的复杂性类 $\mathcal{P}/\text{poly}$ 的一个意外行为蕴含着一致的复杂性类 (PH 属于此类) 的意外行为。

证明: 利用假设条件 (即 $\mathcal{NP} \subset \mathcal{P}/\text{poly}$), 从任意一个集合 $S \in \Pi_2$ 开始, 证明 $S \in \Sigma_2$ 。首

先描述主要证明思想。

不严格地说, $S \in \Pi_2$ 意味着 $x \in S$ 当且仅当对所有的 y , 存在一个 z , 使得关于 (x, y, z) 的某个 (固定的) 多项式时间可验证的条件成立。注意到关于 (x, y) 的剩余条件是 NP-类型的, 从而 (由假设), 可以利用多项式规模电路来验证。这表明, $x \in S$ 当且仅当存在充分的电路 C 使得对所有的 y , 有 $C(x, y) = 1$ 。因此, 我们试着交换全称量词与存在量词的顺序。具体地, 如果我们或者能验证 coNP (或者即使是在 Σ_2 中) 的充分条件, 或者当 $x \notin S$ 且 C 不充分时不造成麻烦, 则最终得到的断言是期望的 Σ_2 -类型。在以下证明中, 可以实现后一种情况, 观察到由假设可以得到 NP-搜索问题 (而不仅仅是 NP-判定问题) 的小规模电路。得到的 (小规模) 电路可以对给定的 (x, y) , 求 (x, y) 的一个 NP-证据 (无论这样的证据是否存在), 并依赖于可以高效验证 NP-证据的正确性这一事实 (习题 3.11 中给出了另一种方法, 用于获得验证电路充分性的 coNP -类型程序)。

上述方法的具体细节: 设 S 为 Π_2 中任意一个集合, 则由命题 3.9, 存在多项式 p 和集合 $S' \in \text{NP}$, 使得 $S = \{x: \forall y \in \{0, 1\}^{p(|x|)}, (x, y) \in S'\}$ 。令 $R' \in \text{PC}$ 为对应于 S' 的证据关系, 即存在一个多项式 p' , 使得 $x' = \langle x, y \rangle \in S'$ 当且仅当存在 $z \in \{0, 1\}^{p'(|x|)}$, 使得 $(x', z) \in R'$ 。于是有

$$S = \left\{x: \forall y \in \{0, 1\}^{p(|x|)} \exists z \in \{0, 1\}^{p'(|\langle x, y \rangle|)} (\langle x, y \rangle, z) \in R'\right\} \quad (3.2)$$

这种方法的工作过程是这样的: 利用从 PC 到 NP 的归约 (见定理 2.10), 定理的假设条件 (即 $\text{NP} \subseteq \text{P/poly}$) 蕴含着存在多项式规模电路, 可以解 R' 的搜索问题。利用这种电路的存在性, 可以得到, 对任意 $x \in S$, 存在小规模电路 C' , 使得对任意 y , 有 $C'(x, y) \in R'(x, y)$ (因为 $\langle x, y \rangle \in S'$, 从而 $R'(x, y) \neq \emptyset$)。另一方面, 对任意 $x \notin S$, 存在一个 y , 使得 $\langle x, y \rangle \notin S'$, 从而对任意电路 C' , 有 $C'(x, y) \notin R'(x, y)$ (由于一个平凡原因, 即 $R'(x, y) \neq \emptyset$)。因此, $x \in S$ 当且仅当存在一个规模为 $\text{poly}(|x| + p(|x|))$ 的电路 C' , 使得对所有的 $y \in \{0, 1\}^{p(|x|)}$, 有 $(\langle x, y \rangle, C'(x, y)) \in R'$ 。令 $V(x, C', y) = 1$ 当且仅当 $(\langle x, y \rangle, C(x, y)) \in R'$, 可以推出 $S \in \Sigma_2$ 。具体细节如下。

首先要解释清楚什么是解搜索问题的多项式规模电路, 并进一步确认对于搜索问题 R' , 这种电路的存在性。在 3.1 节, 我们重点讨论了解判定问题的多项式规模电路。然而, 3.1.1 节给出的定义也可用于解搜索问题的情形, 只要用一个恰当的习惯方法来将 (固定实例的) 长度不同的解编码为固定长度。下一步, 观察到将 PC 到 NP 的归约与 $\text{NP} \subseteq \text{P/poly}$ 的假设相结合, 蕴含了 PC 可以 Cook-归约为 P/poly 。特别地, 这样做暗示着 PC 中任意搜索问题都可由一类多项式规模电路求解。注意到最终得到的能解这类问题的一个 n -比特长实例的电路, 可能是将 (关于 n 的) 多项式个电路组合起来的结果, 其中每个电路可解 m -比特长实例, 这里 $m \in [\text{poly}(n)]$ 。当然, 最终得到的解上述 $R' \in \text{PC}$ 的搜索问题 (长为 n 的实例) 的电路, 其规模上限为 $\text{poly}(n) \cdot \sum_{m=1}^{\text{poly}(n)} \text{poly}(m)$ 。

下面我们证明, $x \in S$ 当且仅当存在一个规模为 $\text{poly}(|x| + p(|x|))$ 的电路, 满足对所有 $y \in \{0, 1\}^{p(|x|)}$, 有 $(\langle x, y \rangle, C'(x, y)) \in R'$ 。注意: $x \in S$ 当且仅当对任意 $y \in \{0, 1\}^{p(|x|)}$, $(x, y) \in S'$,

这意味着存在 $z \in \{0,1\}^{p(|x|)}$, 满足 $(\langle x, y \rangle, z) \in R'$ 。再由上述讨论想到, 存在着解 R' 的搜索问题的多项式规模电路。因此, 当 $x \in S$ 时, 只需利用相应的输入长度为 $|x| + p(|x|)$ 的电路 C' 求解 R' 的搜索问题。事实上, 这个电路 C' 只取决于 $n' = |x| + p(|x|)$, 这个值又取决于 $|x|$, 且对每个 $x' \in \{0,1\}^{n'}$, 有 $(x', C'(x')) \in R'$ 当且仅当 $x' \in S'$ 。因此, 对于 $x \in S$, 存在一个规模为 $\text{poly}(|x| + p(|x|))$ 的电路 C' , 满足对任意 $y \in \{0,1\}^{p(|x|)}$, 有 $(\langle x, y \rangle, C'(x, y)) \in R'$ 。另一方面, 如果 $x \notin S$, 则存在一个 y , 对所有的 z , 有 $(\langle x, y \rangle, z) \notin R'$ 。从而, 在这种情况下, 对每个 C' , 存在一个 y , 使得 $(\langle x, y \rangle, C'(x, y)) \notin R'$ 。由此得到结论, $x \in S$ 当且仅当

$$\exists C' \in \{0,1\}^{\text{poly}(|x|+p(|x|))} \forall y \in \{0,1\}^{p(|x|)} (\langle x, y \rangle, C'(x, y)) \in R' \quad (3.3)$$

观察式 (3.3) 中给出的条件, 会发现它具有期望的形式 (是 Σ_2 型陈述)。具体地, 考虑多项式时间的验证程序 V , 给定 x, y 及对电路 C' 的描述, V 首先计算 $z \leftarrow C'(x, y)$, 并当且仅当 $(\langle x, y \rangle, z) \in R'$ 时接受, 而这个条件可在多项式时间内验证 (因为 $R' \in \text{PC}$)。将可能的电路描述记作 $\langle C' \rangle$, 上述 V 的 (多项式时间) 计算记作 $V(x, \langle C' \rangle, y)$, 并且 $x \in S$ 当且仅当

$$\exists \langle C' \rangle \in \{0,1\}^{\text{poly}(|x|+p(|x|))} \forall y \in \{0,1\}^{p(|x|)} V(x, \langle C' \rangle, y) = 1$$

已经证明了对任意的 $S \in \Pi_2$ 有 $S \in \Sigma_2$, 从而可以得到结论 $\Pi_2 \subseteq \Sigma_2$ 。定理得证 (利用习题 3.9.4)。

本章注释

作为“接受建议机器”^[139]概念的推广, Karp 和 Lipton^[139]给出了 P/poly 类的定义。他们还指出了 P/poly 类与传统的多项式规模电路的等价性及其一致性效果 (命题 3.4)。

多项式时间层级 ($\mathcal{PH}$) 由 Stockmeyer^[213]引入。 $\mathcal{PH}$ 的第三种等价定义 (利用所谓的交替机, alternating machines) 可见于参考文献[52]。

Karp 和 Lipton^[139]发现, 在多项式时间层级下 $\mathcal{NP}$ 不包含于 P/poly 中 (即定理 3.12) 的猜想可能是错误的。这种非一致性与一致的复杂性类之间的有趣的联系, 是我们将 P/poly 和 $\mathcal{PH}$ 放在同一章介绍的主要原因。

习 题

3.1 (P/poly 定义的一种微小变形) 如果把长度小于 n 的串充分地编码为 n -比特串 (如将 $x \in \bigcup_{i < n} \{0,1\}^i$ 编码为 $x01^{n-|x|-1}$), 则利用这一点, 可将电路 (或接受建议的机器) 定义为能处理长度不超过给定界限的输入的设备 (而非只能处理固定长度输入的设备)。证明 P/poly 类在这种变换下保持不变 (且定理 3.6 仍成立)。

3.2 (稀疏集) 集合 $S \subset \{0,1\}^*$ 称为稀疏的, 如果存在多项式 p , 使得对任意 n , 有 $|S \cap \{0,1\}^n| \leq p(n)$ 。

(1) 证明任意稀疏集都属于 P/poly 。注意: 一个稀疏集可能是不可判定的。

(2) 证明一个集合属于 $P/poly$ 当且仅当其可以 Cook-归约到某个稀疏集。

提示: 在第二部分中, 证明必要性时, 将建议序列 $(a_n)_{n \in \mathbb{N}}$ 编码为稀疏集 $\{(l^n, i, \sigma_{n,i}) : n \in \mathbb{N}, i \leq |a_n|\}$, 其中 $\sigma_{n,i}$ 为 a_n 的第 i 比特。证明充分性时, 注意到对于输入 x , 要模拟到集合 S 的 Cook-归约, 只需知道 $S \cap \bigcup_{i=1}^{\text{poly}(|x|)} \{0,1\}^i$ 。

3.3 (建议的层级) 证明对任意两个函数 $l, \delta: \mathbb{N} \rightarrow \mathbb{N}$, 满足 $l(n) < 2^{n-1}$ 且 δ 无界限, 有 P/l 是 $P/(l+\delta)$ 的真子集。

提示: 对任意序列 $\bar{a} = (a_n)_{n \in \mathbb{N}}$, 满足 $|a_n| = l(n) + \delta(n) \leq 2^n$, 考虑 $\bar{a}$ 的编码集合 $S_{\bar{a}}$, 用 $\text{idx}(x)$ 表示 x 在 $\{0,1\}^n$ 中的序号/索引, 当且仅当 a_n 中第 $\text{idx}(x)$ 个比特为 1 (且 $\text{idx}(x) \leq |a_n|$) 时, $x \in S_{\bar{a}} \cap \{0,1\}^n$ 。更多细节见 4.1 节。

3.4 证明 Σ_2 中包含了所有能 Cook-归约到 $\mathcal{NP}$ 的集合。

提示: 这个结论是很显然的, 可直接利用 3.2.2 节中 Σ_2 的定义, 见习题 3.9。另外, 还可利用定理 2.35 的证明思想, 只有将 $\mathcal{NP}$ 与 coNP 断言相结合, 才能构成一个 Σ_2 类型的断言 (还可见于命题 3.11 证明的第二部分)。

3.5 令 $\Delta = \mathcal{NP} \cap \text{coNP}$, 证明 Δ 与可以 Cook-归约为 Δ 的判定问题类相等 (即 $\Delta = P^{\Delta}$)。

提示: 参考定理 2.35 的证明。

3.6 (Π_k 类) 回顾 Π_k 被定义为与 $\text{co}\Sigma_k$ 相等, 后者又定义为与 $\{ \{0,1\}^* \setminus S : S \in \Sigma_k \}$ 相等。

证明对任意自然数 k , 如果存在多项式 p 及多项式时间算法 V , 满足 $x \in S$ 当且仅当

$$\forall y_1 \in \{0,1\}^{p(|x|)} \exists y_2 \in \{0,1\}^{p(|x|)} \forall y_3 \in \{0,1\}^{p(|x|)} \cdots Q_k y_k \in \{0,1\}^{p(|x|)}$$

使得 $V(x, y_1, \dots, y_k) = 1$, 其中当 k 为奇数时, Q_k 为全称量词, k 为偶数时, Q_k 为存在量词, 则判定问题 $S \subseteq \{0,1\}^*$ 属于 Π_k 。

3.7 ($\mathcal{PH}$ 各种层次中的完全问题) 在一个由存在量词开始的量化布尔公式中, 如果至多有 k 次存在量词与全称量词间的转换, 则称该公式为 k -转换的量化布尔公式。例如, $\exists z_1 \exists z_2 \forall z_3 \phi(z_1, z_2, z_3)$ (其中 z_i 为布尔变量), 是 2-转换的。

证明: 对任意 $k \geq 1$, 判定一个 k -转换的量化布尔公式是否有效在 Karp-归约下是 Σ_k -完全的。换言之, 将上述问题记作 kQBF, 要证明 kQBF 属于 Σ_k , 且 Σ_k 中所有问题可以 Karp-归约到 kQBF。

提示: 从 k 为奇数的情形开始。这样可以对最后一个子句中 (归约过程中引入的) 辅助变量的存在量词进行合并。对于偶数 $k > 1$, 首先考虑一个类似的 Σ_k -完全问题, 再考虑其补问题。

3.8 ($\mathcal{PH}$ 与 $\mathcal{AC}^0$ 间的关系) 注意到 $\mathcal{PH}$ 与无界扇入的常数深度电路之间有明显的相似性, 其中存在 (或全称) 量词用 “大的” $\vee$ 门 (或 $\wedge$ 门) 表示, 为了更清楚地表达这种关系, 考虑以下定义。

- 电路类 $\{C_N\}$ 称为强一致的, 如果存在多项式时间算法, 能对关于相应电路结构的询问做出应答。具体地, 对于输入 (N, u, v) , 算法确定 C_N 中由顶点 u 和 v 所代表的门的类型, 以及 u 与 v 之间是否有边相连。如果顶点代表一个终端, 则算法也会指出相应的输入比特 (或输出比特)。注意: 该算法的运行时间为 C_N 规模的多项式。

我们主要考虑多项式规模电路类, 这表示 C_N 的规模是 N 的多项式, 而规模又代表了 C_N 输入的个数。

- 固定多项式 p , 输入 $Z \in \{0,1\}^N$ 的 p -简明表示是一个规模至多为 $p(\log_2 N)$ 的电路 c_Z , 满足对任意 $i \in [N]$, $c_Z(i)$ 等于 Z 的第 i 比特。

- 给定一个强一致性电路类 $\{C_N\}$ 以及固定多项式 p , 对于简明表示的输入求值定义如下: 给定输入 $Z \in \{0,1\}^N$ 的 p -简明表示, 确定是否有 $C_N(Z)=1$ 。

证明如下 $\mathcal{PH}$ 与关于某个强一致的深度有界电路类对简明表示的输入求值之间的关系。

(1) 对任意 k 及每个 $S \in \Sigma_k$, 证明存在一类强一致的、深度为 k 的、具有多项式规模的无界扇入电路, 满足 S 可以 Karp-归约为对简明表示输入的求值 (关于该类电路)。即归约应将实例 $x \in \{0,1\}^n$ 映射为某个输入 $Z \in \{0,1\}^N$ 的 p -简明表示, 使得 $x \in S$ 当且仅当 $C_N(Z)=1$ (注意到 Z 表示为电路 c_Z , 使得 $\log_2 N \leq |c_Z| \leq \text{poly}(n)$, 从而 $N \leq \exp(\text{poly}(n))$)。①

提示: 令 $S \in \Sigma_k$ 并令 V 为如定义 3.8 中的验证算法。即 $x \in S$ 当且仅当 $\exists y_1 \forall y_2 \cdots Q_k y_k$, 其中每个 $y_i \in \{0,1\}^{\text{poly}(|x|)}$, 且满足 $V(x, y_1, \dots, y_k)=1$ 。因此, 对于 $m = \text{poly}(|x|)$ 和 $N = 2^{k \cdot m}$, 考虑固定的电路 $C_N(Z) = \bigvee_{i_1 \in [2^m]} \bigwedge_{i_2 \in [2^m]} \cdots Q'_{i_k \in [2^m]} Z_{i_1 i_2 \cdots i_k}$, 以及对包含 $V(x, \dots)$ 的真值表的输入计算电路 C_N 的问题 (即设 $Z_{i_1 i_2 \cdots i_k} = V(x, i_1, \dots, i_k)$, 其中 $[2^m] = \{0,1\}^m$, 意为 Z 在本质上表示为 x)。② 注意 C_N 的规模为 $O(N)$ 。

(2) 对任意 k 及所有具有无界扇入的, 深度为 k 的多项式规模强一致电路类, 证明相应的对简明表示输入的求值问题或者在 Σ_k 中, 或者在 Π_k 中。

提示: 给定 Z 的一种简明表示, $C_N(Z)$ 的值可以理解为一个含 k 次量词转换的量化布尔公式。该公式对某些由 C_N 的输出到其输入端之间的路径进行量化, 例如, 一个 $\vee$ -门 (或 $\wedge$ -门) 被赋值为 1 当且仅当其孩子结点之一 (或全部) 被赋值为 1。利用 C_N 的一致性条件, 易高效识别一个结点的孩子及其相应的输入比特。输入比特本身的价值也可由 Z 的简明表示高效地计算。

3.9 验证以下事实。

(1) 对任意 $k \geq 0$, 有 $\Sigma_k \subseteq \mathcal{P}^{\Sigma_k} \subseteq \Sigma_{k+1}$ (回顾, 对任意复杂性类 C , $\mathcal{P}^C$ 类表示可以 Cook-归约为 C 中某个集合的类, 特别地, $\mathcal{P}^P = \mathcal{P}$)。

(2) 对任意 $k \geq 0$, $\Pi_k \subseteq \mathcal{P}^{\Pi_k} \subseteq \Pi_{k+1}$ (提示: 对任意复杂性类 C , 有 $\mathcal{P}^C = \mathcal{P}^{\text{co}C}$ 及 $\mathcal{P}^C = \text{co}\mathcal{P}^C$)。

(3) 对任意 $k \geq 0$, 有 $\Sigma_k \subseteq \Pi_{k+1}$ 及 $\Pi_k \subseteq \Sigma_{k+1}$, 从而有 $\mathcal{PH} = \bigcup_k \Pi_k$ 。

(4) 对任意 $k \geq 0$, 如果 $\Sigma_k \subseteq \Pi_k$ (或 $\Pi_k \subseteq \Sigma_k$), 则 $\Sigma_k = \Pi_k$ (提示: 见习题 2.37)。

① 假设 $\mathcal{P} \neq \mathcal{NP}$, 则不可能有 $N \leq \text{poly}(n)$ (因为电路求值所需时间为电路规模的多项式)。

② 注意到 $\mathcal{AC}^0$ 电路在计算上的局限性 (见参考文献 [83, 115]) 蕴含了上述电路 C_N 在对函数计算普通输入 Z 时也具有局限性。更重要的是, 这种局限性对于 $Z = h(Z')$ 也成立, 其中 $Z' \in \{0,1\}^{N^{O(1)}}$ 是一个普通输入, 且 Z 的每个比特或者等于 Z' 中某个固定比特, 或者为其逆。然而, 不幸的是, 这些计算上的局限性似乎不能提供关于具有简明表示的, 输入为 Z 的函数的限制的有用信息 (为得到这种简明表示, 可令 $Z_{i_1 i_2 \cdots i_k} = V(x, i_1, \dots, i_k)$, 其中 V 为固定的多项式时间算法, 且只有 $x \in \{0,1\}^{\text{poly}(\log N)}$ 在变化)。这个基本问题可在“相对化”情况下“解决”, 方法是向 V 提供对任意长度 (大约) 为 $N^{O(1)}$ 的输入的预言访问, 见参考文献 [83]。

3.10 续习题 3.7, 证明如下断言:

(1) SAT 在计算上等价 (在 Karp-归约下) 于 1QBF。

(2) 对任意 $k \geq 1$, 有 $\mathcal{P}^{\Sigma_k} = \mathcal{P}^{k\text{QBF}}$ 。

提示: 证明如果 S 是 C -完全的, 则 $\mathcal{P}^C = \mathcal{P}^S$, 注意: $\mathcal{P}^C \subseteq \mathcal{P}^S$ 可利用 C 到 S 的多项式时间归约, 而 $\mathcal{P}^S \subseteq \mathcal{P}^C$ 可利用 $S \in C$ 。

3.11 (定理 3.12 的另一种证明) 续定理 3.12 证明中的讨论, 利用以下指示, 给出对定理 3.12 的另一种证明方法。

(1) 首先, 证明如果 T 可以下行自归约 (定义见习题 2.13), 则判定 T 的电路的正确性可在 coNP 中判定。具体地, 将 T 的特征函数记作 χ , 证明集合

$$\text{ckt}_\chi \stackrel{\text{def}}{=} \left\{ \langle 1^n, \langle C \rangle \rangle : \forall \omega \in \{0, 1\}^n, C(\omega) = \chi(\omega) \right\}$$

包含于 coNP 中。注意: 可以对 T 不做任何假设, 除了假设 T 是下行自归约的之外。

提示: 利用习题 3.1 中给出的更灵活的定义, 只需验证对任意 $i < n$ 及任意的 i 比特串 ω , $C(\omega)$ 的值等于在得到关于 C 的应答时, 对输入 ω 的下行自归约的输出。因此, 对任意 $i < n$, C 对长为 i 的输入的正确性可由其对长度小于 i 的输入的正确性而得出。当然, 为了验证 C 对于空串 (或对所有具有某种常数长度输入) 的正确性, 可以比较其与空串上的固定 χ 值 (或常数数量串的 χ 值)。

(2) 回顾 SAT 也是下行自归约的, 并且 NP 可以 Karp-归约到 SAT, 从而定理 3.12 是第一部分的一个推论。

提示: 令 $S \in \Pi_2$, $S' \in \text{NP}$ 如定理 3.12 证明中所述。设 f 表示 S' 到 SAT 的一个 Karp-归约, 注意 $S = \left\{ x : \forall y \in \{0, 1\}^{p(|x|)} f(x, y) \in \text{SAT} \right\}$ 。利用 SAT 有多项式规模电路的假设, 注意到 $x \in S$ 当且仅当存在一个规模为 $\text{poly}(|x|)$ 的电路 C , 满足: ① C 可对任意长度不超过 $\text{poly}(|x|)$ 的输入能正确判定 SAT; ② 对任意 $y \in \{0, 1\}^{p(|x|)}$ 有 $C(f(x, y)) = 1$ 。推断 $S \in \Sigma_2$ 。

3.12 续习题 3.2 的第二部分, 考虑可以 Karp-归约为一个稀疏集的集合类。证明该类中包含 SAT 当且仅当 $\mathcal{P} = \text{NP}$ (见参考文献[81])。这里只考虑一种特殊情形, 即稀疏集包含于一个可在多项式时间内判定的稀疏集中 (如后者可能为 $\{1\}^*$, 此时前者可能是任意一个单元素集合)。实际上, 要证明的是以下这个似乎更强的断言:

如果 SAT 可以 Karp-归约为集合 $S \subseteq G$, 满足 $G \in \mathcal{P}$ 且 $G \setminus S$ 为稀疏集, 则 $\text{SAT} \in \mathcal{P}$ 。

利用假设, 可以粗略描述一个多项式时间程序, 来求解 SAT 的搜索问题, 我们将该程序的细节设计留做练习。程序在由输入公式的所有可能部分真值赋值构成的树中进行 DFS,^① 并在已经证明无用的部分真值赋值对应的结点处中止搜索。

提示: 一个重要观察是, 每个内部结点 (通过对原始公式中相应的部分真值赋值进行例示而得到的公式) 被 Karp-归约映射为一个不属于 G 的串 (此时, 可以得到结论, 该子树中不包含可满足的赋值, 并从该结点处返回) 或者 G 中的串。在后一种情况下, 除非已知该串不属于 S , 否则从以该结点为根的子树开始扫描。然而, 一旦要从这个内部结点返回, 我们

① 对一个 n 变量公式, 树的叶子对应着所有可能的 n 比特串, 而对应于 τ 的内部结点是对应于 τ_0 和 τ_1 的结点的父亲结点。

就知道 G 中的相应元素不属于 S ，从而不会扫描以该元素的原像结点为根的子树。还需注意的是，一旦到达某个叶子结点，便可自行检查该结点是否对应着原始公式的一种可满足的赋值。

提示：在分析上述程序时，注意对于输入的 n 变量公式 ϕ ，需要对子树进行扫描的次数至多为 $n \cdot \left| \bigcup_{i=1}^{\text{poly}(|\phi|)} \{0,1\}^i \cap (G \setminus S) \right|$ 。

第 4 章

资源越多功能就越强大吗？

多用电，少费力。
以色列电力公司，20 世纪 60 年代

是否拥有的资源越多，完成的任务也越多呢？回答也许是显然的，但这个显然的（肯定的）回答实际上有一个前提，那就是假设工作者知道其掌握的资源。在这种情况下，当分配到更多资源时，工作者（或计算）确实可以完成更多任务。但是如果该假设不成立，则增加资源也没有什么用处。

在计算复杂性理论中，如果一个算法可以确定为其分配的资源数量，并且确定过程所需资源不超过相应资源量时，称该算法知道为其分配的资源数量。这一条件在所有“合理”的情况下是满足的，但它也并不总成立。这丝毫不奇怪：我们已经知道某些函数是不可计算的，如果利用这些函数来确定资源，则算法就会陷入困境。当然，需要将这种讨论形式化，这正是本章的目的。

内容提要

当使用“良好”的函数来确定算法资源时，用更多的资源确实可以完成更多的任务。然而，当使用“坏”的函数时，资源的增加可能并不起什么作用。这里“良好”的函数是指计算该函数时所需资源没有超过规定数量（如 $t(n) = n^2$ 或 $t(n) = 2^n$ ）的函数。自然地，“坏”的函数不具备这种好的条件。

上述讨论针对着一致的计算模型和“自然的”计算资源，如时间和空间复杂性。因此，我们会得到这样的结论，如存在一些问题，可以在 3 次方的时间内解决，但不能在平方时间内解决。在非一致的计算模型中，并不存在函数“好”与“坏”的问题，并且易区分无限多个不同的电路复杂性等级。

为了区分在一种资源范围可解的问题类与利用更多资源可解的问题类，相关的结论称为层级定理（Hierarchy Theorems）。表明不存在这种区别的结论，从而指出计算能力的增加中存在“缝隙”（或使用这些增加的计算资源的算法中存在“缝隙”）的结论称为缝隙定理（Gap Theorems）。还有一个相关现象称为加速定理（Speed-up Theorems），涉及到某些问题的复杂性无法定义的情形。

附带说明

基于特定资源界限（如 3 次方时间）的一致的复杂性类依赖于计算模型。进一步地，分

离结论（即解某些附加的计算问题“额外”需要多少时间）的紧凑性也依赖于计算模型。在所有合理且得到普遍应用的计算模型下，这种分离都是存在的（如 Cobham-Edmonds 定理）。以下我们将明确区分普遍效应与依赖于特定模型的效应。

内容组织

首先，阐述在非一致的复杂性类中，“更多资源会导致更强大的功能”这一现象。此时，并不存在是否“知道”分配给计算设备的资源数量的问题，因为每个设备都被分配了适于其处理的输入长度所需的资源（见 4.1 节）。然后，在 4.2 节将讨论一致性算法的时间复杂性，就是时间复杂性的层级和缝隙定理，这是本章的主要内容。最后，介绍空间复杂性中的类似结论（见 4.3 节，可以在阅读了 5.1 节之后再回来看这一节）。

4.1 非一致的复杂性层级

使用建议的机器模型（见 1.2.4.2 节和 3.1.2 节）为分离结论提供了一个非常便利的环境。我们特别针对的是形如 P/l 的类，其中 $l: \mathbf{N} \rightarrow \mathbf{N}$ 是任意一个函数（见定义 3.5）。注意：可以借助于一个给定的平凡算法，将函数的真值表（限制到相应输入长度上）作为建议，从而使每个布尔函数都属于 $P/2^n$ 。基于类似算法可得到下述的分离结论。

定理 4.1 对任意两个函数 $l', \delta: \mathbf{N} \rightarrow \mathbf{N}$ ，满足 $l'(n) + \delta(n) \leq 2^n$ ，且 δ 无界限，则 P/l' 是 $P/(l' + \delta)$ 的真子集。

证明： 令 $l \stackrel{\text{def}}{=} l' + \delta$ ，考虑如下接受建议的算法 A ：给定建议 $a_n \in \{0, 1\}^{l(n)}$ 及输入 $i \in \{1, 2, \dots, 2^n\}$ （视为一个 n 比特串），如果 $i \leq |a_n|$ ，则 A 输出 a_n 的第 i 比特，否则输出 0。显然，对任意满足 $|a_n| = l(n)$ 的 $\bar{a} = (a_n)_{n \in \mathbf{N}}$ ，函数 $f_{\bar{a}}(x) \stackrel{\text{def}}{=} A(a_n, x)$ 属于 P/l 。进一步地，不同的序列 $\bar{a}$ 生成不同的函数 $f_{\bar{a}}$ 。我们认为某些 $f_{\bar{a}}$ 不属于 P/l' ，从而得到一种分离。

上述断言的证明需要考虑所有可能的（多项式时间）算法 A' 以及所有满足 $|a'_n| = l'(n)$ 的序列 $\bar{a}' = (a'_n)_{n \in \mathbf{N}}$ 。固定任意一个算法 A' ，我们考虑可以由 $A'(a'_n, \cdot)$ 计算的 n 比特长函数的数量。显然，这个数量不超过 $2^{l'(n)}$ ，从而函数 $f_{\bar{a}}$ 中 A' 可以计算的比例最多为 $2^{-\delta(n)}$ （即使是局限于 n 比特串时）。重要的是，这一点对任意的 n 及 A' 都成立，因此，由接受 l' 长建议的算法计算的函数类所占比例为零。

更严格地，对任意 n ，考虑所有接受建议的算法，其描述长度不超过 $\delta(n) - 2$ （这个长度保证了将所有接受建议的算法都考虑在内）。结合所有可能的长为 l' 的建议序列，这些算法最多可以计算具有 n 比特输入的 $2^{(\delta(n)-2)+l'(n)}$ 个不同函数。后一个数字达不到由 A 在接受 $l(n)$ 长建议时可计算的函数（具有 n 比特输入）数量，即 $2^{l(n)}$ 。 ■

使用布尔电路模型可以得到一个不太紧凑的界限。此时，在考虑能用规模 $s < 2^n$ 的布尔电路计算的 $\{0, 1\}^n$ 上的布尔函数数量时，其上界与下界间的缝隙需要放宽。具体来讲（见习题 4.1），显然这个数量的下界为 $2^{s/O(\log s)}$ ，而一个明显的上界为 $s^{2s} = 2^{2s \log_2 s}$ （利用长为 $l'(n)$ 的建议可计算的函数数量），在定理 4.1 的证明中使用了这两个界限。

4.2 时间层级及缝隙

本节证明在“合理情形”下增加时间复杂度可以解决更多的问题，而在“病态情形”下，即使时间复杂度有了极大增长也不会增加计算能力。本章在引言中暗示，“合理情形”是指可以由算法本身在规定时间复杂度内确定时间界限。

我们强调在上述的“合理情形”下，不是只有自然的计算问题才能增加计算能力。也就是说，与非一致复杂性相似（即定理 4.1），可以引入人造的计算问题来证明层级定理。当然，目前并不知道 $\mathcal{P}$ 类中哪些自然问题在 3 次方时间（或其他多项式时间）内不可解（假设利用双带图灵机）。因此，虽然 $\mathcal{P}$ 中包含了无数层的计算问题，每一次比上一层明显需要更多的求解时间，但我们并不知道对于自然的计算问题的这种分层。相反，目前为止任何被证明在多项式时间内可解的自然问题最终都存在运行时间上限为多项式的算法。

4.2.1 时间层级

注意：4.1 节中的非一致计算设备明确给出了相关的资源上限（如建议长度）。实际上，给出的是资源本身（如具体建议），并且不需要监控对这些资源的使用。与此相反，在设计具有任意时间复杂度 $t: \mathbb{N} \rightarrow \mathbb{N}$ 的算法时，需要确信算法没有超过时间上限。进一步地，当对输入 x 调用算法时，并没有明确给出算法的时间界限 $t(|x|)$ ，一种合理的设计方法是让算法在做其他运算之前先求出这个界限（如 $t(|x|)$ ）。这就要求算法能读取整个输入（见习题 4.3）并在大约 $O(t(n))$ 步内（否则，这个预处理阶段会耗费过多的时间）计算出 $t(n)$ 。后一个要求引出如下定义（与“完全可定时构造”的标准定义相关）（见参考文献[123]中 12.3 节）。

定义 4.2（可定时构造的函数） 函数 $t: \mathbb{N} \rightarrow \mathbb{N}$ 称为可定时构造的，如果存在算法，对于输入 n ，在最多 $t(n)$ 步内输出 $t(n)$ 。

等价地，可以要求映射 $1^n \mapsto t(n)$ 在时间复杂度 t 内计算。但是要注意，上述定义依赖于计算模型，然而相对较“好”的函数计算起来会更快（如可在 $\text{poly}(\log t(n))$ 步内计算），此时将不再依赖于具体的计算模型（指合理且通用的计算模型，如在 Cobham-Edmonds 命题中涉及到的）。例如，在任意合理且通用的计算模型下，诸如 $t_1(n) = n^2$ ， $t_2(n) = 2^n$ 及 $t_3(n) = 2^{2^n}$ 等函数可在 $\text{poly}(\log t_i(n))$ 步内计算。

类似地，对于给定计算模型（由上下文中的理解）以及任意函数 $t: \mathbb{N} \rightarrow \mathbb{N}$ ，用 $D_{\text{TIME}}(t)$ 表示在时间复杂度 t 内可解的判定问题，读者可以看一下习题 4.7，其中称在许多情况下 $D_{\text{TIME}}(t) = D_{\text{TIME}}(t/2)$ 。

4.2.1.1 时间层级定理

以下定理（其中区分了 $D_{\text{TIME}}(t_1)$ 与 $D_{\text{TIME}}(t_2)$ ）中使用双带图灵机模型。此时，会得到用 t_1 与 t_2 关系表示的相当紧凑的层级。我们强调，根据 Cobham-Edmonds 命题，这个结论适用于任意合理且通用的计算模型（可能不那么紧凑）。

教学注释：在定理 4.3 的标准陈述中，对任意可定时构造的函数 t_1 及任意函数 t_2 ，当 $t_2 = \omega(t_1 \log t_1)$ 且 $t_1(n) > n$ 时， $D_{\text{TIME}}(t_1)$ 是 $D_{\text{TIME}}(t_2)$ 的真子集。现在对这个版本稍做弱化，但证明起来更加简单和直观。在证明了这个版本之后，我们给出定理 4.3 标准版本的证明。

定理 4.3 (双带图灵机的时间层级) 对任意可定时构造的函数 t_1 及任意函数 t_2 , 当 $t_2(n) \geq (\log t_1(n))^2 \cdot t_1(n)$ 且 $t_1(n) > n$ 时, $D_{\text{TIME}}(t_1)$ 是 $D_{\text{TIME}}(t_2)$ 的真子集。

证明过程清楚地表明, 在任意模型下均成立的一个类似结论中, 通用机可以在时间 $O(t \log t)$ 内模拟另一个机器的 t 步运行, 函数 $O(\cdot)$ 中的常量依赖于被模拟的机器。在证明定理 4.3 之前, 我们先给出如下推论。

推论 4.4 (在任意合理且通用模型下的时间层级) 对于任意合理且通用的计算模型, 存在一个正多项式 p , 使得对任意可定时计算的函数 t_1 及任意函数 t_2 , 当 $t_2 > p(t_1)$ 且 $t_1(n) > n$ 时, $D_{\text{TIME}}(t_1)$ 是 $D_{\text{TIME}}(t_2)$ 的真子集。

由此可推出, 对任意一个这类模型及任意多项式 t (使得 $t(n) > n$), $\mathcal{P}$ 类中存在一些不属于 $D_{\text{TIME}}(t)$ 的问题。还可推出, $\mathcal{P}$ 是 ε 的真子集, 即使是在“半多项式时间”(即 $D_{\text{TIME}}(q)$, 其中 $q(n) = \exp(\text{poly}(\log n))$) 下也成立。进一步地, 对于任意超多项式函数 q (即 $q(n) = n^{\omega(1)}$), $\mathcal{P}$ 是 $D_{\text{TIME}}(q)$ 的真子集。

推论 4.4 可以直接证明 (不引用定理 4.3)。只需将定理 4.3 的证明思想直接应用于相应计算模型中 (见习题 4.5) 即可。事实上, 这种直接的方法, 其中允许“多项式的松弛” (即 $t_2 > p(t_1)$), 比定理 4.3 中使用多项式一对数因子松弛 (即 $t_2 \geq \tilde{O}(t_1)$) 实现起来更方便。我们还要指出推论 4.4 中的分离结论可以更紧凑——详见习题 4.6。

推论 4.4 的证明: 一个基本事实是, 任意合理且通用的计算模型中的分离结论可以“转化”为其他模型下的类似结论。以下将说明, 这种转化会影响时间上限。令 D_{TIME2} 表示双带图灵机模型下的相应问题类 (回顾 D_{TIME} 表示在其他模型下的相应问题类), 注意 $D_{\text{TIME}}(t_1) \subseteq D_{\text{TIME2}}(t'_1)$ 且 $D_{\text{TIME}}(t_2) \subseteq D_{\text{TIME2}}(t'_2)$, 其中 $t'_1 = \text{poly}(t_1)$ 而 t'_2 被定义为满足 $t_2(n) = \text{poly}(t'_2(n))$, 后面还有两个未详细定义的多项式, 以下分别记作 p_1 和 p_2 , 是在 Cobham-Edmonds 命题中所保证的多项式。另外, t_1 可定时构造的假设蕴含了 $t'_1 = p_1(t_1)$ 也可在双带图灵机模型下定时构造。因此, 对于适当选择的多项式 p (即 $p(p_1^{-1}(m)) \geq p_2(m^2)$), 有

$$t'_2(n) = p_2^{-1}(t_2(n)) > p_2^{-1}(p(t_1(n))) = p_2^{-1}\left(p\left(p_1^{-1}(t'_1(n))\right)\right) \geq t'_1(n)^2$$

由推论中的假设 (即 $t_2 > p(t_1)$) 可知第一个不等式成立, 而由 p 的选择知最后一个不等式成立。利用定理 4.3 (并注意到 $t'_2(n) > t'_1(n)^2$), 可得到严格包含关系 $D_{\text{TIME2}}(t'_1) \subset D_{\text{TIME2}}(t'_2)$ 。结合 $D_{\text{TIME}}(t_1) \subseteq D_{\text{TIME2}}(t'_1)$ 及 $D_{\text{TIME2}}(t'_2) \subseteq D_{\text{TIME}}(t_2)$, 推论得证。 ■

定理 4.3 的证明: 主要思路是构造一个布尔函数 f , 使得所有时间复杂度为 t_1 的机器都无法计算 f 。为此可令每个可能的机器 M 与一个不同输入 x_M 相关 (即 $x_M = \langle M \rangle$), 并保证 $f(x_M) \neq M'(x_M)$, 其中 $M'(x)$ 是对 $M(x)$ 的模拟, 并在 $t_1(|x|)$ 步之后挂起。例如, 可以定义 $f(x_M) = 1 - M'(x_M)$, 注意: 这里使用了 M' 而非 M , 这是为了允许在与 t_1 相关的时间内计算 f 。这里的要点是 M 是与输入 x_M 相关的任意一个机器, 所以 M 不一定在时间 t_1 内运行 (但是, 由构造可知, M' 运行于时间 t_1 内)。

为实现上述思想, 需要高效地将机器与输入相关联, 并且要相对高效地模拟任意机器的 t_1

步运算。以下将说明,这两个要求都是容易满足的。实际上,我们要使用由输入到机器的映射 μ (即 μ 将上面的 x_M 映射到 M) 使得每个机器都在 μ 的值域之内,并且 μ 是易计算的(对于初学者而言,可以假设 μ 就是恒等映射)。因此,由构造可知, $f \notin D_{\text{TIME}}(t_1)$, 现在面临的问题是构造计算 f 的高效算法,即证明 $f \in D_{\text{TIME}}(t_2)$ 。

计算 f 的算法以及 f 的定义(在第一段中已经简要描述)是很直接的:对于输入 x , 算法计算 $t = t_1(|x|)$, 确定相应于 x 的机器 $M = \mu(x)$ (如果这样的机器不存在,则输出一个缺省值), 模拟 $M(x)$ 的 t 步运算, 返回 $1 - M'(x)$ 。注意: $M'(x)$ 表示对 $M(x)$ 的时间截段模拟(即在 t 步之后模拟会挂起), 也就是说, 如果 $M(x)$ 在 t 步之内停机, 则 $M'(x) = M(x)$, 否则, $M'(x)$ 可定义为任意值。因此, 如果 $M = \mu(x)$, 则 $f(x) = 1 - M'(x)$, 否则, (比如)可令 $f(x) = 0$ 。

为了证明 $f \notin D_{\text{TIME}}(t_1)$, 需证明每个时间复杂度为 t_1 的机器都无法计算 f 。给定任意一种这种机器 M , 考虑输入 x_M 使得 $M = \mu(x_M)$, 由于 μ 为满射, 所以这种输入一定存在。现在, 一方面, $M'(x_M) = M(x_M)$ (因为 M 的时间复杂度为 t_1), 另一方面, $f(x_M) = 1 - M'(x_M)$ (由 f 的定义), 从而 $M(x) \neq f(x)$ 。

现在通过分析上述计算 f 的算法的时间复杂度来求 f 的时间复杂度上界。根据 t_1 可定时构造的特性, 并忽略计算 μ 所用的少量时间, 我们重点考虑(对于输入 x) 模拟 M 的 t 步运算需要多少时间。此时, 必须记住, 我们是在双带图灵机模型上分析算法的时间复杂度, 而 M 自身是一个双带图灵机。首先在三带图灵机上实现算法, 然后用双带图灵机来模拟该算法。

在三带图灵机上实现算法时, 显然, 模拟本身需要两个带, 将第三条带用于模拟程序(即保存被模拟机器的代码, 并保持一个运行步骤计数器)。因此, 双带机 M 的每一步运算都可在 $O(|\langle M \rangle|)$ 步内用三带机来模拟。^①这也包括了在模拟过程中保持一个步骤计数器的复杂度(见习题 4.7)。

下面, 需要在双带机上模拟上述的三带机。可以利用这样一个事实(可见参考文献[119]中定理 12.6), 即三带机的 t' 步运算可以用双带机在 $O(t' \log t')$ 步内模拟。因此, f 对于输入 x 的计算的复杂度上界为 $O(T_{\mu(x)}(|x|) \log T_{\mu(x)}(|x|))$, 其中 $T_M(n) = O(|\langle M \rangle| \cdot t_1(n))$ 表示在三带机上模拟双带机 M 的 $t_1(n)$ 步运算需要花费的计算开销(如以上讨论)。

我们得到的分离结论的优势依赖于对映射 μ (由输入到机器的映射)的选择。如果采用简单(恒等)映射(即 $\mu(x) = x$), 则只能对 $t_2(n) = \tilde{O}(n \cdot t_1(n))$ 证明定理的结论, 而非 $t_2(n) = \tilde{O}(t_1(n))$, 因为此时 $T_{\mu(x)}(|x|) = O(|x| \cdot t_1(|x|))$ (注意: 此时 $x_M = \langle M \rangle$ 是对机器 $\mu(x_M) = M$ 的一种描述)。将机器 M 与输入 $x_M = \langle M \rangle 01^m$ 相关联, 其中 $m = 2^{|M|}$, 即可证明该定理。也就是说, 可以利用这样一个映射 μ , 当 $x = \langle M \rangle 01^{2^{|M|}}$ 时, $\mu(x) = M$, 否则, $\mu(x)$ 等于某个固定的机器。此时, 有 $|\mu(x)| < \log_2 |x| < \log t_1(|x|)$, 故 $T_{\mu(x)}(|x|) = O((\log t_1(|x|)) \cdot t_1(|x|))$, 定理得证。 ■

教学注释: 仅仅是将足够长的输入 x_M 与每个机器 M 相关联并不能证明定理 4.3 的标准版

① 这个负载包括了为实现充分操作而搜索 M 的代码, 以及操作本身的效果(也许涉及到一个更大的字母表, 而非模拟器使用的字母表)。

本, 因为这样不能消除 $T_{\mu(x)}(|x|)$ 中附加的无上限因子 (即与 $t_1(|x|)$ 相乘的因子 $|\mu(x)|$)。注意: 要求这个因子 (至少) 是可计算的, 因此并不能解释标准版本中出现的通用的 ω -记号 (见参考文献[119]中定理 12.9)。事实上, 标准版本证明中采用了另外一种不同方法^①。

技术性说明: 定理 4.3 的证明中将每个可能的机器 M 与输入 x_M 相关联, 并定义了使 M 对输入 x_M 的计算出错的计算问题。上述关联是很灵活的: 可利用任意一个能高效计算并产生足够压缩的由输入到机器的满射。具体地, 使用映射 μ , 满足当 $x = \langle M \rangle 01^{2^{|M|}}$ 时, $\mu(x) = M$, 否则, $\mu(x)$ 等于某个固定的机器。这里要指出, 如果重新定义 μ , 使得当 $\langle M \rangle 01^{2^{|M|}}$ 是 x 的前缀时, $\mu(x) = M$ (否则, $\mu(x)$ 等于某个固定的机器), 则每个机器都会对无穷多个输入计算出错。另外, 还要说明的是, 与定理 4.3 的证明相反, 定理 1.5 的证明使用了一个严格定义的由输入到机器的映射 (即当 $x = \langle M \rangle$ 时, $\mu(x) = M$)。

思考: 对角化

最后的说明中强调了一个事实, 即定理 4.3 的证明仅仅是定理 1.5 证明的一个更复杂的版本。两个证明过程都提到了通用 (Universal) 函数, 在定理 4.3 的证明中 (非显式地) 定义了函数在 $(\langle M \rangle, x)$ 的取值为 $M'(x)$, 其中 $M'(x)$ 表示模拟 $M(x)$ 并在 $t_1(|x|)$ 步之后挂起。^②实际上, 两个证明都使用了上述函数的“对角线”, 而在定理 4.3 的证明中这一点没有明确定义。就是说, 点 x 的对角线函数 (Diagonal Function) 值, 记作 $d(x)$, 等于通用函数在点 $(\langle \mu(x) \rangle, x)$ 的值。这实际上是一种定义模型, 因为没有详细规定对函数 μ 的选择。事实上, 令 $\mu(x) = x$ 对应于表示通用函数的矩阵中“真正”的对角线, 但是定义 μ 为其他的一一映射也能得到通用函数的“类似对角线”。无论使用何种方法, 函数 f 的定义中都要求对任意 x , 有 $f(x) \neq d(x)$, 这保证了任意时间复杂度为 t_1 的机器都无法计算 f , 从而证明的重点是构造计算 f 的算法 (当然, 该算法的时间复杂度大于 t_1)。定理 4.3 的证明中讨论了如何选择 μ 使计算 f 的时间复杂度最小, 而在定理 1.5 的证明中, 只需确保 f 是可计算的即可。

4.2.1.2 通用计算加速的不可行性

时间层级定理 (定理 4.3) 暗示了通用机的计算过程是不能明显加速的。即如果对于输入 x , 机器 M 在 t 步内停止并输出串 y , 则 $\mu'(\langle M \rangle, x, t) \stackrel{\text{def}}{=} y$; 当 M 对输入 x 的计算超过了 t 步时, $\mu'(\langle M \rangle, x, t) \stackrel{\text{def}}{=} \perp$ 。注意: $\mu'(\langle M \rangle, x, t)$ 的值可以在 $\tilde{O}(|x| + |\langle M \rangle| \cdot t)$ 步内求出。以下将证明, 定理 4.3 蕴含了计算该值 (即 $\mu'(\langle M \rangle, x, t)$) 所需步骤数量不会明显减少。

类似结果在任意合理且通用的计算模型中也成立 (见推论 4.4)。特别地, μ' 不可能在多项式时间内计算 (因为输入 t 是二进制表示的)。事实上, 可以证明, 不存在多项式时间算法, 能判定 M 对于输入 x 是否在 t 步内停机 (即集合 $\{(\langle M \rangle, x, t) : \mu'(\langle M \rangle, x, t) \neq \perp\}$ 的成员判定) 不

① 在标准版本的证明中, f 的定义没有参考 $M(x_M)$ 的 $t_1(|x_M|)$ 步执行, 但是参考了模拟 $M(x_M)$ 的结果, 模拟 (即用于计算 f 的算法) 的总步数为 $t_2(|x_M|)$ 。这保证了 f 属于 $D_{\text{TIME}}(t_2)$, 并且“将问题推向”证明 f 不属于 $D_{\text{TIME}}(t_1)$ 。同时还解释了为什么假设 t_2 (而非 t_1) 是可定时构造的。为了解决上述问题, 观察到每个机器 (时间复杂度为 t_1) 对足够长输入的计算都能被充分地模拟。因此, 只需要将每个 M 与不相交的无限多个输入集相关联, 并确保 M 对这些输入都出错即可。

② 显然, 在定理 1.5 的证明中 $M' = M$ 。

属于 $\mathcal{P}$, 见习题 4.8。

证明: (用反证法) 假设对任意机器 M , 给定输入 x 及 $t > |x|$, $\mu'(\langle M \rangle, x, t)$ 的值可在 $o(t/\log^2 t)$ 步内求出, 其中记号 “ o ” 中隐藏了一个可能依赖于 M 的常量。我们将证明这个假设中蕴含了对任意可定时构造的 t_1 和 $t_2(n) = t_1(n) \cdot \log^2 t_1(n)$, 有 $D_{\text{TIME}}(t_2) = D_{\text{TIME}}(t_1)$, 从而 (显然) 与定理 4.3 矛盾。

考虑任意一个可定时构造的函数 t_1 (满足 $t_1(n) > n$) 及任意集合 $S \in D_{\text{TIME}}(t_2)$, 其中 $t_2(n) = t_1(n) \cdot \log^2 t_1(n)$ 。设 M 为判定 S 成员的机器, 其时间复杂度为 t_2 。考虑如下算法: 对于输入 x , 首先计算 $t = t_1(x)$, 再计算 (并输出) $\mu'(\langle M \rangle, x, t \log^2 t)$ 。由于 t_1 是可定时构造的, 第一步的计算可以在 t 步内完成, 而由假设可知, 第二步计算也可在 t 步完成, 从而, S 可以在时间 $D_{\text{TIME}}(2t_1) = D_{\text{TIME}}(t_1)$ 内判定, 故 $D_{\text{TIME}}(t_2) = D_{\text{TIME}}(t_1)$, 与定理 4.3 矛盾。由此得出假设是不合理的, 定理得证。 ■

4.2.1.3 非确定时间的层级定理

类似于 D_{TIME} , 对于给定的计算模型 (由上下文确定) 及任意函数 $t: \mathbb{N} \rightarrow \mathbb{N}$, 用 $N_{\text{TIME}}(t)$ 表示可由某种时间复杂度为 t 的非确定型机器接受的集合类。事实上, 这个定义扩展了传统的 $\mathcal{NP}$ 类 (如定义 2.7)。类似于之前对 $\mathcal{NP}$ 类的定义, 集合 $S \subseteq \{0,1\}^*$ 属于 $N_{\text{TIME}}(t)$, 如果存在线性时间算法 V 满足以下两个条件。

(1) 对任意 $x \in S$, 存在 $y \in \{0,1\}^{t(|x|)}$ 使得 $V(x, y) = 1$ 。

(2) 对任意 $x \notin S$ 及任意 $y \in \{0,1\}^*$, 有 $V(x, y) = 0$ 。

注意: 以上两个定义并不相同, 但是在足够强的模型中 (如双带图灵机), 它们最多相差一个对数因子 (见习题 4.10)。层级定理本身类似于确定时间中的层级定理, 除了在此要求 $t_2(n) \geq (\log t_1(n+1))^2 \cdot t_1(n+1)$ (而非 $t_2(n) \geq (\log t_1(n))^2 \cdot t_1(n)$)。

定理 4.6 (双带图灵机的非确定时间层级) 对任意可定时构造的单调非降函数 t_1 及任意函数 t_2 , 当 $t_2(n) \geq (\log t_1(n+1))^2 \cdot t_1(n+1)$ 且 $t_1(n) > n$ 时, $N_{\text{TIME}}(t_1)$ 是 $N_{\text{TIME}}(t_2)$ 的真子集。

证明: 不能直接利用定理 4.3 的证明, 因为其中定义的布尔函数 f 要求能判定是否存在 M 的一种计算, 能在 $t_1(|x_M|)$ 步内接受输入 x_M 。而在本定理中, M 为非确定机器, 所以解决上述判定问题 (包括了 “是” 和 “否” 两种回答) 的唯一方法是尝试所有的 $(2^{t_1(|x_M|)})$ 个输入。^① 但是这会让 f 属于 $D_{\text{TIME}}(2^{t_1})$ 而非 $N_{\text{TIME}}(\tilde{O}(t_1))$, 所以需要想别的办法。

我们将每个 (非确定的) 机器 M 与一个大的串 (可以看作是整数) 区间相关联, 记作 $I_M = [\alpha_M, \beta_M]$, 并使不同的区间不相交, 故易于判定任意一个串 x 属于哪个区间。对每个 $x \in [\alpha_M, \beta_M - 1]$, 定义 $f(x) = 1$ 当且仅当存在 M 的非确定计算, 能在 $t_1(|x|) \leq t_1(|x| + 1)$ 步内接受输入 $x^{\text{def}} = x + 1$ 。因此, 如果 M 的时间复杂度为 t_1 , 且 (非确定地) 接受 $\{x: f(x) = 1\}$, 则或者 M 接受区间 I_M 内的所有串, 或者 M (非确定地) 不接受 I_M 中的任何串, 因为此时必有

① 事实上, 可以非确定地在 $\tilde{O}(t_1(|x_M|))$ 步内识别回答 “是”, 但是无法识别回答 “否”。

M 非确定地接受 x 当且仅当其非确定地接受 $x' = x + 1$ 。所以剩下的问题就是要处理 M 对于 I_M 保持不变的情况, 此时对 $f(\beta_M)$ 取值的定义开始发挥作用: 定义 $f(\beta_M) = 0$ 当且仅当存在 M 的非确定计算, 能在 $t_1(|\alpha_M|)$ 步内接受输入 α_M 。我们将选择 β_M 比 α_M 足够大, 从而可以尝试 M 对所有输入 α_M 的计算, 具体细节如下。

首先简要地重述一下 $f: \{0,1\}^* \rightarrow \{0,1\}$ 的定义, 重点考虑输入位于某个区间 I_M 的情况。定义布尔函数 A_M , 使得 $A_M(z) = 1$ 当且仅当存在 M 的非确定计算, 能在 $t_1(|z|)$ 步内接受输入 z , 从而对于 $x \in I_m$, 有

$$f(x) = \begin{cases} A_M(x+1) & , x \in [\alpha_M, \beta_M - 1] \\ 1 - A_M(\alpha_M) & , x = \beta_M \end{cases}$$

接下来构造能接受集合 $\{x: f(x) = 1\}$ 的非确定机器。假设对于输入 x , 易确定相应于 x 所在区间 $[\alpha_M, \beta_M]$ 的机器 M 。^①区分如下两种情况。

(1) 对于输入 $x \in [\alpha_M, \beta_M - 1]$, 这个非确定机器模拟 M 对于输入 $x' = x + 1$ 的 $t_1(|x'|)$ 步非确定型计算, 并做出相应的判决 (当且仅当模拟接受时我们构造的机器才接受)。事实上 (与定理 4.3 的证明相同), 模拟需要的时间为 $(\log t_1(|x+1|))^2 \cdot t_1(|x+1|) \leq t_2(x)$ 。

(2) 对于 $x = \beta_M$, 我们构造的机器只尝试 M 对于输入 α_M 的所有 $2^{t_1(|\alpha_M|)}$ 种执行, 并以适当的方式判断, 即在 $M(\alpha_M)$ 的 $2^{t_1(|\alpha_M|)}$ 种可能的执行中, 模拟每种执行的 $t_1(|\alpha_M|)$ 步, 当且仅当所有被模拟的执行都不接受 α_M 时接受 β_M 。注意: 机器的这一部分是确定的, 且可归结为模拟 M 的 $T_M \stackrel{\text{def}}{=} 2^{t_1(|\alpha_M|)} \cdot t_1(|\alpha_M|)$ 步。若选择的区间 $[\alpha_M, \beta_M]$ 恰当 (如 $|\beta_M| > T_M$), 则模拟的步数 (即 T_M) 小于 $|\beta_M| \leq t_1(|\beta_M|)$, 从而模拟 M 的 T_M 步所需时间为 $(\log_2 t_1(|\beta_M|))^2 \cdot t_1(|\beta_M|) \leq t_2(|\beta_M|)$ 。

因此, 以上构造的非确定机器时间复杂度为 t_2 , 从而 f 属于 $N_{\text{TIME}}(t_2)$, 接下来证明 f 不属于 $N_{\text{TIME}}(t_1)$ 。

用反证法, 假设时间复杂度为 t_1 的某个非确定型机器 M 接受集合 $\{x: f(x) = 1\}$, 即对任意 x , 有 $A_M(x) = f(x)$, A_M 的定义如前所述 (即 $A_M(x) = 1$ 当且仅当存在 M 的非确定型计算, 能在 $t_1(|x|)$ 步内接受 x)。重点讨论区间 $[\alpha_M, \beta_M]$ 的情况, 对任意 $x \in [\alpha_M, \beta_M]$, 有 $A_M(x) = f(x)$, 这 (结合 f 的定义) 蕴含了对任意 $x \in [\alpha_M, \beta_M - 1]$, 有 $A_M(x) = f(x) = A_M(x+1)$ 且 $A_M(\beta_M) = f(\beta_M) = 1 - A_M(\alpha_M)$, 从而得出矛盾 (因为已经有 $A_M(\alpha_M) = \dots = A_M(\beta_M) = 1 - A_M(\alpha_M)$)。■

4.2.2 时间缝隙及加速

与定理 4.3 相反, 存在一些函数 $t: \mathbb{N} \rightarrow \mathbb{N}$ 使得 $D_{\text{TIME}}(t) = D_{\text{TIME}}(t^2)$ (或者甚至

① 例如, 可将所有的串划分为连续区间, 第 i 个区间, 记作 $[\alpha_i, \beta_i]$, 对应于第 i 个机器, 并对 $T_1(M) = 2^{2t_1(m)}$, 有 $\beta_i = 1^{T_1(|\alpha_i|)}$ 且 $\alpha_{i+1} = 0^{T_1(|\alpha_i|)+1}$ 。注意: $|\beta_i| = T_1(|\alpha_i|)$, 因此 $t_1(|\beta_i|) > t_1(|\alpha_i|) \cdot 2^{t_1(|\alpha_i|)}$ 。

$D_{\text{TIME}}(t) = D_{\text{TIME}}(2^t)$ 。当然, 这些函数是不能定时构造的(因此, 上述事实与定理 4.3 不矛盾)。之所以出现这种现象, 是因为对于这种函数 t , 不存在时间复杂度大于 t 而小于 t^2 (或 2^t) 的算法。

定理 4.7 (时间缝隙定理) 对任意非降的可计算函数 $g: \mathbb{N} \rightarrow \mathbb{N}$, 存在非降的可计算函数 $t: \mathbb{N} \rightarrow \mathbb{N}$ 使得 $D_{\text{TIME}}(t) = D_{\text{TIME}}(g(t))$ 。

在上述例子中, $g(m) = m^2$ (或 $g(m) = 2^m$)。由于我们感兴趣的是缝隙较大的情形(即 g 为超多项式函数), 因此在这里计算模型的影响可以忽略(只要是合理且通用的模型即可)。

证明: 考虑枚举所有可能的算法(或机器), 其中包括对某些输入不停机的机器(因为我们不可能列举出对每个输入均停机的所有机器)。令 t_i 表示第 i 个算法的时间复杂度, 即如果第 i 个机器对某个 n 比特输入不停机, 则 $t_i(n) = \infty$, 否则, $t_i(n) = \max_{x \in \{0,1\}^n} \{T_i(x)\}$, 其中 $T_i(x)$ 表示第 i 个机器计算输入 x 时所需的计算步骤数。

这里的基本思想是在定义 t 时, 使得所有的 t_i 都不会位于 t 和 $g(t)$ 之间, 从而不存在时间复杂度位于 t 与 $g(t)$ 之间的算法。直观地讲, 如果 $t_i(n)$ 有限, 则可以定义 t 使得 $t(n) > t_i(n)$, 这样就保证了 $t_i(n) \notin [t(n), g(t(n))]$, 而当 $t_i(n) = \infty$ 时, 可令 $t(n)$ 取任意有限值(因为此时必有 $t_i(n) > g(t(n))$)。因此, 对任意的 m 和 n , 可以定义 $t(n)$ 使得对任意 $i \in [m]$, 有 $t_i(n) \notin [t(n), g(t(n))]$ (如可令 $t(n) = \max_{i \in [m]: t_i(n) \neq \infty} \{t_i(n)\} + 1$)。^①这样就得到了定理的一个弱化版本, 其中的函数 t 是可计算的, 或者是非降的。将 t 修改为非降函数是很容易的(如令 $t(n) = \max(t(n-1), \max_{i \in [m]: t_i(n) \neq \infty} \{t_i(n)\} + 1)$), 所以这里真正的难点在于要让 t 可计算。

问题在于我们的目标是令 t 可计算, 而对于给定的 n 并不能判断 $t_i(n)$ 是否有限。然而, 这个判断并非必需的: 对于 $t(n)$ 的任意候选值 v , 只需判断是否有 $t_i(n) \in [v, g(v)]$ 即可, 这可以利用第 i 个机器在至多 $g(v)+1$ 步(对每个 n 比特串)内判断。也就是说, 在考虑第 i 个机器时, 应该找到一个值 v , 使得或者 $v > t_i(n)$ 或者 $g(v) < t_i(n)$ (这包括了 $t_i(n) = \infty$ 的情况)。可以从 $v = v_0$ 开始(如可令 $v_0 = n+1$), 逐渐增加 v 直到 $v > t_i(n)$ 或 $g(v) < t_i(n)$ 。关键在于如果 $t_i(n)$ 是无限的, 则在模拟了 $2^n \cdot (g(v_0)+1)$ 步之后可以输出 $v = v_0$, 否则, 将在模拟了至多 $\sum_{j=v_0}^{t_i(n)} 2^n \cdot j$ 步之后, 得到一个安全的值 $v > t_i(n)$ 。注意: 在这里要处理的是所有机器, 我们得到以下设置 $t(n)$ 的程序。

令 $\mu: \mathbb{N} \rightarrow \mathbb{N}$ 为任意一个无界且可计算的函数(如 $\mu(n) = n$), 从 $v = n+1$ 开始, 不断增加 v 的值直到对任意 $i \in \{1, 2, \dots, \mu(n)\}$, 或者 $t_i(n) < v$ 或者 $t_i(n) > g(v)$ 。可通过计算 $\mu(n)$ 及 $g(v)$ 的值来验证这个条件, 并模仿前 $\mu(n)$ 个机器对所有 n 比特串计算的 $g(v)+1$ 步。这个过程将设置 $t(n)$ 为第一个满足条件的 v 值, 然后停止(图 4.1 描述了为 $t(n)$ 寻找一个“好”的 v 值的过程)。

为证明上述过程对任意的 n 都停机, 考虑集合 $H_n \subseteq \{1, 2, \dots, \mu(n)\}$, H_n 为所有对 n 长输入都停机的机器下标集合, 则上述过程在到达 $v = \max(T_n, t(n-1)) + 2$ 之前一定停机, 其中

① 不失一般性, 可以假设对任意 n , 有 $t_1(n) = 1$, 要做到这一点, 只需在枚举时让第一个机器在执行了一步之后总停机。

$T_n = \max_{i \in H_n} \{t_i(n)\}$ (事实上, 可能在 $v \leq T_n$ 时就停机, 但这种情况只出现在 $g(v) < T_n$ 时)。

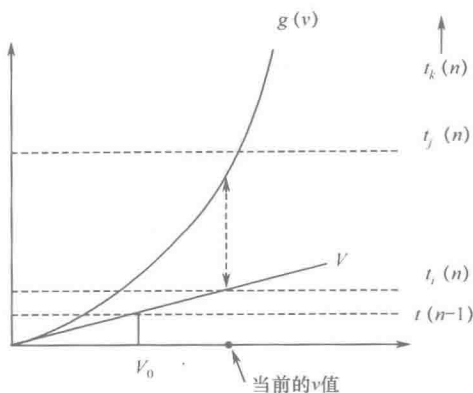


图 4.1 缝隙定理——确定 $t(n)$ 值

最后, 对于上述函数 t , 证明 $D_{\text{TIME}}(t) = D_{\text{TIME}}(g(t))$ 成立。事实上, 考虑任意一个 $S \in D_{\text{TIME}}(g(t))$, 并假设第 i 个算法能在不超过 $g(t)$ 的时间内判定 S , 即对任意 n , 有 $t_i(n) < g(t(n))$, 从而 (由 t 的构造), 对任意满足 $\mu(n) \geq i$ 的 n , 有 $t_i(n) < t(n)$ 。这就证明了第 i 个算法可在至多时间 t 内对几乎所有输入判定 S 。将这个算法与处理例外输入的“查找表”(Look-up Table) 机器相结合, 可得出结论 $S \in D_{\text{TIME}}(t)$, 从而, 定理得证。■

说明: 上述证明中定义的函数 t 在大于 $g(t)$ 的时间内可计算。具体而言, 上述程序在时间 $\tilde{O}(2^n \cdot g(t(n)) + T_g(t(n)))$ 内计算 $t(n)$ (以及 $g(f(n))$), 其中 $T_g(m)$ 表示对于输入 m , 计算 $g(m)$ 所需步骤数。

加速定理

可以认为, 定理 4.7 称某些时间复杂性类 (即定理中的 $D_{\text{TIME}}(g(t))$) 会衰减到更低层的类 (即 $D_{\text{TIME}}(t)$)。一个概念上相关的现象是不存在最优算法 (即使是较为温和的优化) 的问题, 即求解这些 (“病态”) 问题的任何算法都可以明显提速, 从而, 无法定义这些问题的复杂度 (即解该问题的最优算法的复杂度)。以下的显著加速定理不应与图灵机定义中人为制造的线性加速定理相混淆 (见习题 4.4)。^①

定理 4.8 (时间加速定理) 对任意可计算的 (以及超线性的) 函数 f , 存在一个确定的集合 S 。满足若 $S \in D_{\text{TIME}}(t)$, 则 $S \in D_{\text{TIME}}(t')$, 其中 t' 满足 $g(t'(n)) < t(n)$ 。

· 如果令 $g(n) = n^2$ (或 $g(n) = 2^n$), 则上述定理的含义是对任意 t , 如果 $S \in D_{\text{TIME}}(t)$, 则 $S \in D_{\text{TIME}}(\sqrt{t})$ (或 $S \in D_{\text{TIME}}(\log t)$)。注意: 理 4.8 可以应用任意 (常数) 次, 这意味着无法合理地估计 S 的成员判定问题。相反, 在某些重要的情形中, 确实存在解计算问题的最优算法。具体来讲, 解 NP 中公正搜索问题的算法不能加速 (见定理 2.33), 通用机器的计算也不能加速 (见定理 4.5)。

我们在这里省略了定理 4.8 的证明, 但是要对证明中使用集合的复杂度做出说明。证明 (见

^① 注意: 在定理 4.3 的证明中已经暗示了线性加速现象, 其中的模拟负载取决于模拟机的描述长度。

参考文献[123]的 12.6 节)中构造了属于 $D_{\text{TIME}}(t') \setminus D_{\text{TIME}}(t'')$ 的集合 S , 其中 $t'(n) = h(n - O(1))$, 且 $t''(n) = h(n - \omega(1))$ 。这里 $h(n)$ 表示 g 对于输入 2 迭代 n 次后的输出 (即 $h(n) = g^{(n)}(2)$, 其中 $g^{(i+1)}(m) = g(g^{(i)}(m))$ 且 $g^{(1)} = g$)。集合 S 的构造要满足对任意 $i > 0$, 存在 $j > i$ 及一个算法, 可在时间 t_i 内判定 S , 但不能在时间 t_j 内判定, 这里 $t_k(n) = h(n - k)$ 。

4.3 空间层级和缝隙

在空间复杂性中, 也存在着分别类似于定理 4.3 和定理 4.7 的层级和缝隙定理。事实上, 对于空间受限机器的高效空间模拟, 比对时间受限机器的高效时间模拟要容易, 得到的结论也更清晰 (证明也更简单)。特别是在 (以下将给出的) 分离结论中, 这一点尤为明显, 该结论是最优的一个结论 (在相应的线性加速结论中, 见习题 4.12)。

在阐述分离结论之前, 需要先介绍一些预备知识。读者可以参考 1.2.3.5 节中空间复杂度的定义 (以及第 5 章中的更多讨论)。在时间复杂性情形中, 我们考虑特定的计算模型, 但是得到的结论在其他合理且通用的模型中也成立。具体地, 我们考虑三带图灵机, 因为需要指定两个特殊的带作为输入和输出带。对任意函数 $s: \mathbb{N} \rightarrow \mathbb{N}$, 将在空间复杂度 s 内可求解的判定问题类记作 $D_{\text{SPACE}}(s)$ 。类似于定义 4.2, 称函数 $s: \mathbb{N} \rightarrow \mathbb{N}$ 是限定空间构造的, 如果存在一个算法, 对于输入的 n , 能输出 $s(n)$, 且在计算中至多利用工作带上 $s(n)$ 个存储空间。实际上, 类似于 $s_1(n) = \log n$, $s_2(n) = (\log n)^2$ 及 $s_3(n) = 2^n$ 的函数, 可在空间 $O(\log s_i(n))$ 内计算。

定理 4.9 (三带图灵机的空间层级) 对任意限定空间构造的函数 s_2 及任意函数 s_1 , 如果 $s_2 = \omega(s_1)$ 且 $s_1(n) > \log n$, 则 $D_{\text{SPACE}}(s_1)$ 是 $D_{\text{SPACE}}(s_2)$ 的真子集。

定理 4.9 类似于定理 4.3 的传统版本 (不是本章给出的版本), 细节留作习题 (见习题 4.13)。

本章注释

本章内容在时间上早于 NP-完全性理论的提出及 P-vs-NP 问题开始占据核心地位的时间。该领域 (即未来的复杂性理论) 在早期尚未发展成为一个独立的研究对象, 其研究角度可以归结为两个经典理论: 可计算理论 (及递归函数) 以及形式语言理论。然而, 我们认为, 本章所提出的结论是有趣的, 原因有两个: 其一, 前面说过, 这些结论针对的自然问题是在何种条件下能使更多的计算资源发挥作用; 其二, 这些结论实际上代表了在计算复杂性的 “一般” 问题中可以得到的结论类型, 这些问题会涉及到任意的资源界限 (如对任意 t_1 和 t_2 , 讨论 $D_{\text{TIME}}(t_1)$ 与 $D_{\text{TIME}}(t_2)$ 之间的关系)。

注意到, 与本章讨论的 “一般” 问题相反, 本书其余章节中涉及到的 P-vs-NP 问题及与之相关的问题并非 “一般” 的, 它们针对着特殊的类 (用以解释自然的计算问题)。进一步地, 虽然时间和空间复杂度关于层级和缝隙的表现是类似的, 但在其他方面却有所不同。感兴趣的读者可以参考 5.1 节和 5.3 节。

至于本章的具体内容, 我们简略指出联系最紧密的资料来源。层级定理 (如定理 4.3) 由

Hartmanis 和 Stearns^[114]证明。缝隙定理(如定理 4.7)由 Borodin^[47]证明(常称为 Borodin 缝隙定理)。Blum^[38]开发了复杂性量度的公理化方法,他还证明了相应的加速定理(如定理 4.8,常被称为 Blum 加速定理)。参考文献[123]第 12 章中提供了所有上述内容的传统表述,还介绍了一些相关的技术(如“翻译引理”)。

习 题

4.1 设 $F_n(s)$ 表示 $\{0,1\}^n$ 上可由规模为 s 的电路计算的不同布尔函数个数。证明对任意 $s < 2^n$, 有 $F_n(s) \geq 2^{s/O(\log s)}$, 以及 $F_n(s) \leq s^{2s}$ 。

提示:任意布尔函数 $f: \{0,1\}^l \rightarrow \{0,1\}$ 可由规模为 $s_l = O(l \cdot 2^l)$ 的电路计算。因此,对任意 $l \leq n$, 有 $F_n(s_l) \geq 2^{2^l} > 2^{s_l/O(\log s_l)}$ 。另一方面,规模为 s 的电路数量小于 $2^s \cdot \binom{s^2}{s}$, 其中第二个因子表示电路中输入到任意一个门的所有可能“门对”的数量。

4.2 (可使计算加速的建议)对任意定时构造的函数 t , 证明 $D_{\text{TIME}}(t^2) \setminus D_{\text{TIME}}(t)$ 中存在集合 S , 可利用线性长度的建议在线性时间内判定(即 $S \in D_{\text{TIME}}(l)/l$, 其中 $l(n) = O(n)$)。

提示:从集合 $S' \in D_{\text{TIME}}(T^2) \setminus D_{\text{TIME}}(T)$ 开始, 其中 $T(m) = t(2^m)$, 考虑集合 $S = \{x0^{2^{|x|}-|x|} : x \in S'\}$ 。

4.3 在任意合理的计算模型下(并假设输入长度未明确给定(与定义 10.10 不同)), 证明任意亚线性时间复杂度的算法实际上具有常数时间复杂度。

提示:考虑是否存在无限多个串构成的集合 S , 在对任意输入 $x \in S$ 调用算法时, 算法可以读出 x 的全部值。注意:如果 S 是无限的, 则算法不可能具备亚线性时间复杂度, 并证明如果 S 是有限的, 则算法具有常数时间复杂度。

4.4 (图灵机的线性加速)证明任意可由时间复杂度为 t 的双带图灵机求解的问题, 也可由另一个时间复杂度为 t' 的双带图灵机求解, 其中 $t'(n) = O(n) + (t(n)/2)$ 。

提示:考虑使用一个更大字母表的机器, 能对原始图灵机的常数个(记作 c)符号编码, 从而能在 $O(1)$ 步内模拟原始图灵机的 c 步, 其中 O -记号中的常数是一个通用常数(独立于 c)。注意: $O(n)$ 项表示将二元输入转化为新机器工作字母表的预处理过程(将 c 个输入比特编码为字母表中的一个符号)。因此, 单带图灵机中的类似结论似乎要求这一项变成 $O(n^2)$ 。

4.5 (推论 4.4 的直接证明)利用定理 4.3 的证明思想, 直接证明推论 4.4。进一步地, 证明如果机器 M (在当前模型下)的 t 步运算可由相应通用机的 $g(|M|, t)$ 步来模拟, 则推论 4.4 对任意 $t_2(n) \geq g(\log n, t_1(n))$ 都成立。

提示:函数 $f \notin D_{\text{TIME}}(t_1)$ 的定义恰如定理 4.3 证明中所给出的, 这里 D_{TIME} 表示当前计算模型下的复杂性类。在该模型中限定 f 的时间复杂度上界时, 令 $T_M(n)$ 表示为模拟 M 的 $t_1(n)$ 步所需步骤数, 并注意 $T_M(n) = g(|M|, t_1(n))$, 且 $f \in D_{\text{TIME}}(T')$, 其中 $T'(n) = \max_{x \in \{0,1\}^n} \{T_{\mu(x)}(n)\}$ 。

4.6 (更紧凑的推论 4.4) 证明在任意合理且通用的计算模型下, 对任意常数 $\varepsilon > 0$ 及任意“良好”的函数 t (例如, 或者对任意常数 $c \geq 1$, 有 $t(n) = n^c$, 或者对任意常数 $c' > 0$, $t(n) = 2^{c'n}$), $D_{\text{TIME}}(t)$ 是 $D_{\text{TIME}}(t^{1+\varepsilon})$ 的真子集。

提示: 用反证法, 对函数 $f(k) = k^{1+\varepsilon}$, 假设 $D_{\text{TIME}}(t) = D_{\text{TIME}}(f \circ t)$, 证明对任意常数 i , 有 $D_{\text{TIME}}(t) = D_{\text{TIME}}(f^i \circ t)$, 其中 f^i 表示 f 的 i 次迭代, 从而与推论 4.4 矛盾。注意到, 为了证明 $D_{\text{TIME}}(t) = D_{\text{TIME}}(f \circ t)$ 蕴含着 $D_{\text{TIME}}(f^{i-1} \circ t) = D_{\text{TIME}}(f^i \circ t)$ (对任意常数 i), 需要处理“填充问题”(即 n 比特输入被编码为 m 比特的输入, 使得 $t(m) = (f^{i-1} \circ t)(n)$, 并且事实上 $n \mapsto m = (t^{-1} \circ f^{i-1} \circ t)(n)$ 应该在时间 $t(m)$ 内计算)。

4.7 (常数均摊时间计步器) 计步器是一个算法, 在输入中规定了其运行的步骤数。实际上, 这样一个算法运行的步骤数可能会更多, 但是在处理了输入中规定的若干个“信号”后将停机, 这些信号定义为进入(或离开)(算法的)一个指定状态。计步器可以与任何程序并行运行, 这是为了在(另一程序)运行预定步数之后挂起。证明存在一个简单的判定机器, 对于输入 n , 在收到 n 个信号且运行了 $O(n)$ 步后停机。

提示: 实现直接算法, 并分析其“均摊”时间复杂度。

4.8 ($\varepsilon \setminus \mathcal{P}$ 中的一个自然的集合) 续定理 4.5, 证明集合 $\{(\langle M \rangle, x, t) : u'(\langle M \rangle, x, t) \neq \perp\}$ 属于 $\varepsilon \setminus \mathcal{P}$, 其中 $\varepsilon \stackrel{\text{def}}{=} \bigcup_c D_{\text{TIME}}(e_c)$ 且 $e_c(n) = 2^{cn}$ 。

4.9 (EXP-完全性) 续习题 4.8, 证明 EXP 中每个集合都可 Karp-归约到集合 $\{(\langle M \rangle, x, t) : u'(\langle M \rangle, x, t) \neq \perp\}$ 。

4.10 证明 4.2.1.3 节中给出的 N_{TIME} 的两种定义相差一个对数因子。注意: V 具有线性(而非多项式)时间复杂度这个条件的重要性。

提示: 在利用 V 的验证程序来模拟非确定机器时, 对于非确定地选择的“证据”串 y 的编码满足, $|y|$ 比原始机器的运行步骤数稍大。具体地, $|y| = O(t \log t)$, 其中 t 表示原始机器的运行步骤数, 允许在线性时间(即 $|y|$ 的线性函数)内模拟后一种运算。

4.11 续定理 4.7, 证明对任意可计算的函数 $t' : \mathbb{N} \rightarrow \mathbb{N}$ 及每个非降的可计算函数 $g : \mathbb{N} \rightarrow \mathbb{N}$, 存在一个非降的可计算函数 $t : \mathbb{N} \rightarrow \mathbb{N}$ 满足 $t > t'$ 且 $D_{\text{TIME}}(t) = D_{\text{TIME}}(g(t))$ 。

4.12 续习题 4.4, 利用 4.3 节给出的对于空间的标准定义, 叙述并证明空间复杂度的线性加速结论(注意: 这个结论在 5.1.1 节定义的“二元空间复杂度”意义下并不成立)。

4.13 证明定理 4.9。作为热身, 先证明定理 4.3 的空间复杂度版本。

提示: 注意用一台机器在空间高效地模拟另一台机器, 比时间高效的类似模拟更容易。

4.14 (空间缝隙定理) 续定理 4.7, 叙述并证明空间复杂度的缝隙定理。

第5章

空间复杂性

随着门门的开启，地平线的两个门被打开了。

菲利普·格拉斯，阿赫那吞，前奏

要衡量算法的效率，除了计算步骤数这个基本量度之外，计算使用的临时存储空间也是一个重要方面。此外，在某些环境中，空间甚至是比时间更稀缺的资源。

除了探索空间复杂性的本质特征之外，对空间复杂性的研究也为研究时间复杂度提供了新的视角。例如，与 $NP \neq coNP$ 这样一个普遍猜测相反，我们将看到类似的空间复杂性类（如 NL 类）对于取补运算是封闭的（如 $NL = coNL$ ）。

内容提要

本章研究计算的空间复杂性，主要讨论两种极端情况。第一种情况是具有对数空间复杂度的算法，我们视其为本质上使用了最小临时存储空间的算法，在这里“最小”是一种直观（但不太准确的）感觉，而且与多项式时间相比，对数空间复杂度的要求似乎更严格。第二种情况是多项式空间复杂度算法，对这类算法的要求比多项式时间复杂度算法要低得多。事实上，本书涉及到的所有计算任务几乎都能用多项式空间的算法来处理（如 $PSPACE$ 类中几乎包含了本书涉及到的所有复杂性类）。

首先考虑对数空间复杂度算法，这种算法可用于解各种搜索和判定问题，构造问题之间的归约，并得出布尔电路的强一致性概念。这一部分的重点是（无向）图漫游的对数空间算法。

然后介绍非确定性的计算，重点是复杂性类 NL ，可用判定（有向）图的有向连通性这一问题来诠释。这一部分的重点是证明了 $NL = coNL$ ，这个等式可以用有向非连通性到有向连通性之间的对数空间归约来解释。

最后简单介绍 $PSPACE$ 类，证明可满足的量化布尔公式是 $PSPACE$ -完全的（在多项式时间归约下）。我们指出该证明与证明 $N_{SPACE}(s) \subseteq D_{SPACE}(O(s^2))$ 之间的相似性。

我们强调，与时间复杂性情况相同，本章的主要结论在任意合理的计算模型下都成立^①。事实上，在适当的定义下，空间复杂性甚至比时间复杂性更健壮。为简单起见，本章仍采用图灵机模型。

内容组织

空间复杂性与时间复杂性完全不同，在研究中主要采用了异于传统的思维方式。5.1 节讨

^① 仅有的例外出现在习题 5.3 和习题 5.14 中，其中涉及到交叉序列（Crossing Sequence）的概念。习题证明中使用这个概念时假定了机器以串行方式扫描其存储设备。相反，我们强调在各种即时配置中并不使用这样一种机器模型。

论了一些相关问题，然后介绍对数空间复杂度（见 5.2 节），以及相应的非确定版本（见 5.3 节）。最后讨论多项式空间复杂度（见 5.4 节）。

5.1 预备知识及相关问题

我们先介绍空间复杂性研究中几个重要习惯（见 5.1.1 节）。显然，5.1.1 节是阅读本章其余内容的基础（相反，相对不太重要的 5.1.2 节可以忽略）。然后讨论各种相关问题，重点是空间复杂性与时间复杂性的区别（见 5.1.3 节）。特别地，读者要注意组合引理（5.1.3.1 节）、相关归约（5.1.3.3 节）以及 5.1.3.2 节中的类似结论（即 $D_{\text{SPACE}}(s) \subseteq D_{\text{TIME}}(2^{O(s)})$ ）。最后，在 5.1.4 节，通过讨论电路求值的空间复杂性，在电路规模与空间复杂度之间建立联系。

5.1.1 几个重要的习惯性表达

空间复杂性考虑计算任务所需要的临时存储空间（即计算机内存）。由于我们主要讨论使用存储量为输入长度亚线性函数的计算任务，因此，采用的计算模型应该能区分用于计算的存储空间和用于存储输入或最终输出的存储空间，这一点非常重要。就是说，不能将输入和输出本身也计入到计算所需空间中，因此，要使用特殊的、不同于内存的设备来处理输入输出。另一方面，也必须确保输出设备不被滥用为工作空间（这种情况不予考虑）。这就形成一个习惯，输入设备（如多带图灵机中指定的输入带）是只读的，而输出设备（如多带图灵机中指定的输出带）是只写的。根据这种习惯，在定义空间复杂性时，只考虑其他（存储）设备（如多带图灵机中的工作带）。

给定一种具体的计算模型（如多带图灵机），用 $D_{\text{SPACE}}(s)$ 表示在空间复杂度 s 内可解的判定问题。搜索问题的空间复杂度定义是类似的。具体来讲，空间复杂度的标准定义（见 1.2.3.5 节）是指对于每个输入需要扫描的工作带上存储单元数量。然而，我们更倾向于另一种定义即二元空间复杂度，其中对实际的存储空间有更精确的计量。具体地，计算的二元空间复杂度是指可以在这些存储单元中保存的比特数，从而为存储单元数增加了一个乘法因子，该因子是机器的工作带符号集中元素个数的对数。^①

上述两种定义没有本质区别，只相差了一个通常会忽略的常数因子。尽管如此，除了概念上的准确性以外，使用二元空间复杂度便于讨论某些技术细节（因为使用二元空间复杂度时，“可能的即时配置”数量有明确的上界，而其与标准定义之间的关系依赖于该机器本身）。在这种应用中，我们还会将机器的有限状态计入其空间复杂度。另外，为简单起见，还假设机器在扫描输入带时，扫描范围不会超出输入界限，其中输入界限用特殊的符号表示。^②

我们强调（只读的）输入带（或设备）上的个别位置会被多次访问。在许多亚线性空间算法中这是必需的（因为这些算法需要不止一次地扫描输入，否则就无法将输入复制到存储设备中）。相反，在只写的输出带上（向同一位置）多次写入则不是必需的，事实上，取消这个规定引起的代价也很小（见习题 5.2）。

① 注意：与时间复杂性不同，线性加速（如习题 4.12）实际上并不会节省空间资源。使用更强大的硬件（如有更大工作字母表的图灵机）确实可以减少计算所需时间，但是将空间划分为更大的块（即使用更大的存储单元）并不会显著影响计算中实际使用的空间。在介绍两种机器的二元空间复杂度时，我们将解释这一事实。

② 习题 5.1 中暗示的，这个自然的假设不会造成什么损失。

总结：以下列出上述的几个习惯，如前所述，前两个习惯十分重要，而其余则是出于技术上的考虑（但是确实便于说明问题）。

(1) 空间复杂性不考虑输入与输出设备。

(2) 输入设备是只读的，输出设备是只写的。

(3) 通常只考虑算法的二元空间复杂度，设机器 M 使用字母表 Σ ，有限状态集 Q ，其标准空间复杂度为 S_M ，则其二元空间复杂度定义为 $(\log_2 |Q|) + (\log_2 |\Sigma|) \cdot S_M$ （这里 S_M 表示 M 在计算中使用的临时存储单元数量）。

(4) 假设机器不会扫描输入带上超出输入界限的地方。

(5) 假设机器不会向输出带上同一位置多次写入（即向输出设备中每个存储单元最多写入一次）。

5.1.2 有用的最少计算空间

要牢记我们的主要目标之一是识别 $\mathcal{P}$ 类的自然子类，此时，需考虑有意义的计算所需的最小空间是什么。注意：正则集（Regular Set，见参考文献[123]中第2章）可利用常数空间图灵机判定，这也是该机器的唯一功能（如可见参考文献[123]中2.6节）。一种诱人的说法是亚对数空间机器并不比常数空间机器更加有用，因为似乎不可能为机器分配亚对数的空间。之所以产生这种错误直觉是由于这样一个假设，即分配非常数空间时需要明确地计算出输入长度，而这个过程又需利用对数空间来实现。然而，这种假设是错误的：（具有适当形式的）输入自身可用于确定其长度（以及/或允许的空间数量）。^①事实上，对于 $l(n) = \log \log n$ ， $D_{\text{SPACE}}(O(l))$ 是 $D_{\text{SPACE}}(O(1))$ 一个恰当的超集，见习题5.3。另一方面，双重对数（Double-logarithmic）空间实际上是比较常数空间更有用的最小空间（见习题5.4），即对于 $l(n) = \log \log n$ ，有 $D_{\text{SPACE}}(o(l)) = D_{\text{SPACE}}(O(1))$ 。

尽管有事实表明，某些非平凡的任务可以在亚对数空间复杂度内完成，但我们要深入研究的最小空间复杂性类仍是对数空间（见5.2节）。以下将看到，几种最基本的计算现象都属于这一类。

附带说明

我们要强调对任何算法而言，过于简单的假设（即为了使用非常数空间而要求明确计算出输入长度）都会导致错误的结论。这说明了“看似合理”（但实际上不合理）的错误假设的危险性。无论何时，在寻找不可能的结论和/或复杂性下界时，必须意识到这种危险的存在。

5.1.3 时间与空间

空间复杂性完全不同于时间复杂性，需要使用不同的研究模式。一个突出的例子是下面将要讨论的算法组合。

5.1.3.1 两个组合引理

与时间不同，空间可以重复利用，但是，另一方面，计算的中间结果也不能无条件地记录。以下的组合引理体现了这两个相互矛盾的现象。

引理 5.1（简单组合） 令 $f_1: \{0,1\}^* \rightarrow \{0,1\}^*$ 和 $f_2: \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ 分别为可在空间

^① 事实上，要使这种方法起作用，应该能够检查输入形式错误的情形（并在亚对数空间内完成这种检查）。

s_1 和 s_2 内计算的函数^①, 则函数 $f(x) \stackrel{\text{def}}{=} f_2(x, f_1(x))$ 可在空间 s 内计算, 且

$$s(n) = \max(s_1(n), s_2(n + l(n))) + l(n) + \delta(n)$$

其中 $l(n) = \max_{x \in \{0,1\}^n} \{l(f_1(x))\}$ 且 $\delta(n) = O(\log(l(n) + s_2(n + l(n)))) = o(s(n))$ 。

当 l 较小时, 引理 5.1 是有用的, 但在多数情况下, $l \gg \max(s_1, s_2)$, 此时, 后面的组合引理将更有用。

证明: 实际上, 计算 $f(x)$ 时, 先求出并保存 $f_1(x)$, 再重复利用空间 (在计算 $f_1(x)$ 时所使用的空间) 来计算 $f_2(x, f_1(x))$ 。这就解释了 $s(n)$ 中的最大值项, 即 $\max(s_1(n), s_2(n + l(n)))$, 这是计算本身所需的 (这部分空间被重复利用), 而空间 $l(n)$ 用于保存中间结果 (即 $f_1(x)$)。 $s(n)$ 中其余的项体现了算法的实现细节。具体来讲, 同一存储设备即用于保存 $f_1(x)$, 也作为 f_2 的计算空间。这意味着在这两部分中都需要记录所处的位置 (即 (计算 f_2 的) 算法在 $f_1(x)$ 中的位置及在其自身工作空间中的位置) (在引理 5.2 证明的结尾将有进一步的讨论)。最后一项 $\delta(n)$ 表示对两个算法进行仿真时所产生的空间负载。

引理 5.2 (仿真组合) 设 f_1 、 f_2 、 s_1 、 s_3 、 l 和 f 如引理 5.1 中所定义, 则 f 可在空间 s 内计算, 且满足

$$s(n) = s_1(n) + s_2(n + l(n)) + O(\log(n + l(n))) + \delta(n)$$

其中 $\delta(n) = O(\log(s_1(n) + s_2(n + l(n)))) = o(s(n))$ 。

图 5.1 描绘了各种组合过程 (其中还表明最直接地组合并不能节省空间)。

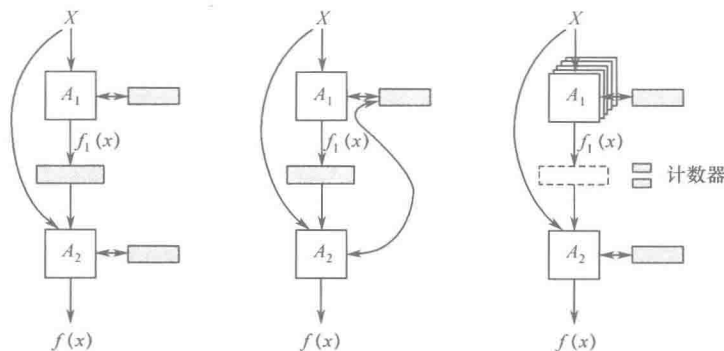


图 5.1 空间受限计算的三种组合方法。左边的图为平凡组合 (只是调用 A_1 和 A_2 而不节省空间), 中间的图为简单组合 (引理 5.1), 右边的图为仿真组合 (引理 5.2)。所有图中用实心矩形表示已分配的存储空间。虚线矩形表示虚拟存储设备

证明: 主要思想是为避免保存 $f_1(x)$ 的中间结果, 在 f_2 的计算中, 根据需要逐个计算 $f_1(x)$

① 这里 (以及整个第 5 章) 为简单起见, 假设所有的复杂性界限都是单调非降的。(正文中) 另一个不太准确的地方是, 我们以一种非标准的方式来阐述计算 f_2 的算法的复杂性。回想在标准习惯下, 算法的复杂性应该用其输入长度来表示, 在这里输入是一个对 (x, y) , 可编码为长 $|x| + |y| + 2\log_2|x|$ 的串 (而非长为 $|x| + |y|$ 的串)。另一个习惯是在阐述计算的复杂性时, 应该同时使用输入中两部分的长度 (即 $s: \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$, 而非 $s: \mathbf{N} \rightarrow \mathbf{N}$), 但我们也没有这样做。

中间结果中的比特。也就是说,不是从计算 $f_1(x)$ 开始,而是在还没有得到相关输入中某些比特(即 $f_1(x)$ 中的比特)时直接计算 $f(x, f_1(x))$,而这些缺少的比特将会根据要求出(以及重复求出)。具体细节如下。

设 A_1 和 A_2 分别为假设中所保证(分别用于计算 f_1 和 f_2)的算法,^①则对输入 $x \in \{0,1\}^n$,调用算法 A_2 (来计算 f_2)。算法 A_2 在调用时有一个虚拟的输入,而在仿真其每一步时需要提供相关的输入比特。因此,我们要在想象中的(虚拟的)输入带上记录 A_2 的位置。如果 A_2 要读输入的第 i 比特, $i \in [n+l(n)]$,当 $i \leq n$ 时,直接由 x 中读出相应的比特,否则,需要调用 $A_1(x)$ 。在调用 $A_1(x)$ 时,为其提供一个虚拟的输出带,逐比特地读出其输出,但并不保存这些值,而是在数到第 $(i-n)$ 个输出比特时,将其传给 A_2 (作为 $\langle x, f_1(x) \rangle$ 的第 i 个比特)。

注意到,在调用 $A_1(x)$ 时,挂起了 A_2 的执行,但保持其当前配置,从而,一旦得到需要的输入比特,便可继续执行 A_2 。因此,需要将计算 A_2 和计算 A_1 的空间分开。另外,还需要分配独立的空间来保存上面所说的计数器(即利用 $\log_2(n+l(n))$ 比特来保存由 A_2 读出的当前输入比特的位置,并用 $\log_2(l(n))$ 比特来保存调用 A_1 时产生的输出比特的索引)。

最后一个(麻烦的)问题是在描述组合算法时需要两个存储设备,一个用于仿真 A_1 的计算,另一个用于仿真 A_2 的计算。问题不在于在两个设备间分配(用于组合算法的)存储空间,而在于算法中需要两个指针(两个存储设备各有一个)。相反,一个(“好的”)组合结论应该给出只使用一个存储设备和一个指针的算法(如 A_1 和 A_2)。事实上,为得到这样的算法,可以将两个原始指针保存于内存之中,而 $s(n)$ 表达式中的 $\delta(n)$ 项就表示这个额外的存储空间。

思考:引理 5.2 证明中提出的算法是比较耗时的:它需要一而再、再而三地重复计算 $f_1(x)$ (只要在计算 A_2 时需要调用其输入的第二部分,就要计算 $f_1(x)$)。然而,在这里,我们的目标是节省空间而非时间(而这两个目标有可能是矛盾的(如可见参考文献[59]中 4.3 节))。

5.1.3.2 一个明显的上界

算法的时间复杂度上限是其空间复杂度的指数函数。这是由于算法可能的即时“配置”数量存在上限(如定理 5.3 的证明中所指),以及如果计算中有两次经过同一配置,则其必陷入死循环。

定理 5.3 如果算法 A 的二元空间复杂度为 s ,并且 A 对任意输入停机,则其时间复杂度 t 满足 $t(n) \leq n \cdot 2^{s(n)+\log_2 s(n)}$ 。

注意:当 $s(n) = \Omega(\log n)$ 时,因子 n 可被 $2^{O(s(n))}$ 吸收,所以只需写成 $t(n) = 2^{O(s(n))}$ 。事实上,本章(以及本书的大部分内容)只考虑对每个输入都停机的算法(进一步的讨论可见于习题 5.5)。

证明:证明涉及(计算中)即时配置的概念。在开始之前,读者需要了解这一概念可能有不同的定义,每种定义都与当前应用相关。在所有定义中,均规定计算的下一步由可变信息与某些固定信息共同确定。在此次证明中,固定算法 A 和输入 x ,将存储设备内容(如图

^① 为了简化问题,我们假设算法 A_1 永远不会在其只写的输出带上(向同一位置)重复写入。如习题 5.2 所述,为了说明这个假设的合理性,需要的额外代价为 $O(\log l(n))$ 。另外,当前证明也直接体现了习题 5.2 中的思想。

灵机的工作带以及其有限状态)及机器在输入设备和存储设备上的位置作为变量。因此, $A(x)$ 的即时配置中包含这 3 个对象(即存储设备内容以及两个位置), 它们可被编码为长 $l(|x|) = s(|x|) + \log_2 |x| + \log_2 s(|x|)$ 的二进制串。^①

一个重要论断是 $A(x)$ 的计算不可能经过同一配置两次, 否则, 就会无数次地经过该配置, 而这意味着计算将永远不会停止。注意到, 即时配置与固定信息(即 A 和 x) 确定了下一步计算, 因此上述论断是合理的。从而, 在 $A(x)$ 的计算中第一次经过配置 γ 时, 无论(在第 i 步之后)发生什么事情, 在第二次经过 γ (第 i 步之后)都会重复发生。

由上述论断可以得到结论: A 对于输入 x 的计算最多包含 $2^{l(|x|)}$ 步, 否则, 同一配置会在计算中重复出现(这与停机假设矛盾)。定理得证。 ■

5.1.3.3 空间受限归约的细节

引理 5.1 和引理 5.2 足以用来分析空间受限计算中多到一归约的效果(函数 f 到函数 g 的多到一归约定义为一个映射 π , 满足对任意 x 有 $f(x) = g(\pi(x))$)。^②

(1) (在引理 5.1 的思想中) 如果 f 可以由一个多到一映射归约为 g , 归约过程在空间 s_1 内计算, 且 g 在空间 s_2 内计算, 则 f 可在空间 s 内计算, 满足 $s(n) = \max(s_1(n), s_2(l(n))) + l(n) + \delta(n)$, 其中 $l(n)$ 表示归约作用于 n 比特串时, 其像的最大长度, 而 $\delta(n) = O(\log(l(n) + s_2(l(n)))) = o(s(n))$ 。

(2) (在引理 5.2 的思想中) 设 f 和 g 如上所述, 则 f 可在空间 s 内计算, 使得 $s(n) = s_1(n) + s_2(l(n)) + O(\log l(n)) + \delta(n)$, 其中 $\delta(n) = O(\log(s_1(n) + s_2(l(n)))) = o(s(n))$ 。

注意: 由定理 5.3, 有 $l(n) \leq 2^{s_1(n) + \log_2 s_1(n)} \cdot n$ 。要强调的是 l 的上限不是 s_1 本身(如在时间受限计算中的类似情况), 而是 $\exp(s_1)$ 。

当讨论通用的(空间受限)归约时, 情况会更加复杂, 特别是当归约需要发出非常数次的询问时。一个基本问题是要对通用归约(即预言机)的空间复杂度进行定义。在标准定义中, 询问和应答的长度并不计入空间复杂度, 但是归约中的询问(或对其应答)被写入到一个专用设备(或从中读取), 该设备对于归约是只写的(或只读的)(并对调用的预言是只读的(或只写的))。注意: 这些习惯与处理输入和输出的习惯类似(并符合上述空间受限的多到一归约的定义)。

上述习惯足以用于定义通用的空间受限归约。从它们出发, 还可得到某些情况下一些有趣的组合结论(如发出单个询问的归约, 或更一般地, 发出非适应性询问的情形)。但是对于通用归约, 要得到组合结论很困难, 通用归约可能发出多个适应性的询问(即这些询问依赖于对前面询问的应答)。下面我们将看到, 在这种情况下, 必须对每个询问及/或应答长度的上限利用初始输入的长度进行限定。

教学注释: 剩余的讨论相对较前沿且粗略(对本章其他内容的学习而言并不是必需的)。

注意: 算法的复杂性来自于预言机以及(实现预言的)实际算法的组合, 该算法依赖于

① 这里利用了 s 为二元空间复杂度(而非标准的空间复杂度)的事实, 见 5.1.1 节中总结的第 3 项。

② 实际上, 这是引理 5.1 和引理 5.2 的一种特殊情形(令 $f_1 = \pi$ 及 $f_2(x, y) = g(y)$)。然而, 针对这种特殊情形的结论要优于调用相应引理所得到的结论(即将 s_2 应用于 $l(n)$ 而非 $n + l(n)$)。

预言机发出的询问长度。例如,上述针对单询问归约的组合算法,其空间复杂度中含 $s_2(l(n))$ 项(其中 $l(n)$ 表示询问长度)。一般来说,第一个询问的长度上限是预言机空间复杂度的指数函数,但是后续的询问不一定如此,除非使用某种附加的强制规定。例如,考虑一种归约,对于输入 x 及预言 f 使得 $|f(z)| = 2|z|$,调用预言 $|x|$ 次,每次用上一次得到的应答作为询问,这个归约使用了常数空间,但是应答的长度超过了输入长度的指数倍,而任意常数空间归约的第一个询问的长度是输入长度的线性函数。要解决这个问题,可以对空间受限归约允许发出的询问长度设置明确上限。例如,可以令所有询问的长度上限为第一个询问的长度(即空间复杂度为 s 的归约只能发出长度不超过 $2^{s(n)+\log_2 s(n)} \cdot n$ 的询问)。

有了上述的习惯(或限制),我们可以考虑一般的空间受限归约与预言的空间受限实现的组合。具体而言,称一个归约是 (l, l') -限定的,如果对于输入 x ,所有的预言询问长度至多为 $l(|x|)$,而相应应答的长度至多为 $l'(|x|)$ 。在简单组合下(引理 5.1),结论仍有效,而对引理 5.2 的仿真组合,结论不成立(因为这个结论比较弱)。

(1) 根据引理 5.1,我们断言,如果问题 Π 可在空间 s_1 内计算,问题 Π' 可在空间 s_2 内计算,当给定对于 Π' 的 (l, l') -限定预言调用时, Π 可在空间 s 内计算,其中

$$s(n) = s_1(n) + s_2(l(n)) + l(n) + l'(n) + \delta(n)$$

$$\delta(n) = O\left(\log(l(n) + l'(n) + s_1(n) + s_2(l(n)))\right) = o(s(n))$$

利用一个简单仿真可以证明这个断言,其中为归约(即预言机)本身,对询问及应答设备的模仿,以及解 Π' 的算法分配了独立的空间。然而在这里要注意,在运行解 Π' 的算法时,不能重复利用归约空间,因为在得到应答之后,就要继续计算归约。附加的 $\delta(n)$ 项针对着预言机中的各种指针,当调用解 Π' 的算法时,就需要保存这些指针(参考引理 5.2 证明的最后一段)。

习题 5.7 中提出了一个相关的组合结论。组合中避免了存储当前的预言询问(但是要保存相应的应答),其中 $s(n) = O(s_1(n) + s_2(l(n)) + l'(n) + \log l(n))$,这里 $l(n) < 2^{O(s_1(n))}$ 表示 $s(n) = O(s_1(n) + s_2(l(n)) + l'(n))$ 。

(2) 再回到引理 5.2 的证明,我们会发现更严重的问题。具体而言,注意重新计算第 i 个询问的应答需要重新计算该询问本身,不同于引理 5.2,它并非归约的输入,而是依赖于对前面询问的应答求出,这些应答也需要重新计算。因此,仿真过程所需要的空间至少是询问次数的线性函数。

注意: 我们不应期望得到一个通用的组合结论(即上述的第一项),其中 $s(n) = F(s_1(n), s_2(l(n))) + o(\min(l(n), l'(n)))$, F 为任意函数。对这个事实的一种解释是,空间复杂度为 s 的任意计算都可以由向常数空间可解问题的归约来模仿,这里的归约是常数空间且为 $(2s, 2s)$ -限定的。

非适应性的归约

在非适应性归约情形下,组合是十分容易的。不严格地说,由这种归约发出的询问不依赖于以前的询问。在空间受限计算中这一概念没有直接定义。而在时间受限计算中,非适应性的归约被看作由两个算法构成:询问生成算法,用于生成询问序列;求值算法,给定输入及(来自预言机的)应答序列,产生实际的输出。此时的归约可看作是调用询问生成算法(并

记录下生成的询问序列), 发出指定询问 (并记录下得到的应答), 最后对应答序列调用求值算法。应用这种定义时会产生一个问题, 即如何描述具有较小空间复杂度的非适应性归约。为解决这一问题, 可以为上述 (询问和应答) 序列分配特殊的存储设备, 并假设这些设备只有这一种用途。细节见习题 5.8。注意到, 非适应性可以克服上面的大部分困难。特别是, 在非适应性归约中, 每个询问的长度上限是归约空间复杂度的指数 (正如单询问归约的情形)。另外, 将这种归约与实现预言的算法相结合, 并不比单询问归约与预言算法的结合更复杂。因此, 正如习题 5.8 所指出的, 如果 Π 可以通过空间复杂度为 s_1 的非适应性归约归约到 Π' , 其中发出的询问长度至多为 l , 且 Π' 可在空间 s_2 内解决, 则 Π 可在空间 s 内解决, 这里 $s(n) = O(s_1(n) + s_2(l(n)))$ (事实上, $l(n) < 2^{O(s_1(n))} \cdot n$ 恒成立)。

到判定问题的归约

对到判定问题的归约进行组合也很容易, 因为此时归约发出的每个询问的长度上限为空间复杂度的指数 (见习题 5.10)。因此, 应用习题 5.7 中的半简单 (Semi-naive) 组合结论 (上面第一项中提及) 是很吸引人的。从而, 在提供了向某个可在空间 s_2 内解决的判定问题的预言访问时, 如果问题 Π 可在空间 s_1 解决, 则 Π 也可在空间 s 内解决, 其中 $s(n) = O(s_1(n) + s_2(2^{s_1(n) + \log(n \cdot s_1(n))}))$ 。事实上, 如果归约中每个询问的长度上限为 l , 则可以使用 $s(n) = O(s_1(n) + s_2(l(n)))$ 。然而, 这些结论的作用很有限, 因为似乎很难构造从搜索问题到判定问题的小空间归约 (见 5.1.3.4 节)。

5.2.4.2 节中讨论了空间受限归约的另一种定义。这个定义更复杂, 约束条件也更多, 但在某些情况下, 它允许使用递归组合, 此时的运算代价小于上述的组合结论。

5.1.3.4 搜索与判定

在时间复杂性中, 我们重点讨论的是判定问题, 因为搜索问题可以高效地归约为判定问题。但不幸的是, 在 (小的) 空间复杂性下, 这些归约 (如定理 2.10 和命题 2.15 中所使用的归约) 并不适用。回顾这些归约通过向判定问题发出询问来扩展当前保存的解的前缀, 因此其空间复杂度下限为解的长度, 这样就不再具有小的空间复杂度了。

鉴于上述讨论, 对搜索问题空间复杂性的研究不能“简化为”对判定问题空间复杂性的研究。因此, 当我们用大部分篇幅讨论判定问题的同时, 也会关注相应的搜索问题。事实上, 在许多情况下, 判定问题的研究思路可以用于研究相应的搜索问题 (如习题 5.17)。

5.1.3.5 复杂性层级和缝隙

第 4 章中讲过, 更多的空间将允许更多的计算 (见定理 4.9), 条件是空间受限的函数在充分意义下是“良好的”。事实上, 对空间复杂度层级和缝隙的证明比时间复杂性中的类似证明更简单, 因为对空间受限算法进行模仿将更容易 (见 4.3 节)。

5.1.3.6 联合的并发时-空复杂度

前面提到, 如果一个计算至少需要对数空间复杂度, 则其所需的时间上限总是空间复杂度的指数函数 (见定理 5.3)。因此, 多项式-对数空间复杂度是多项式时间的扩展, 从而, 有必要定义一个类, 其中包含所有可利用多项式-对数空间的多项式时间算法求解的判定问题。这个类记作 SC , 实际上是 P 的自然子类 (且包含了 5.2.1 节中定义的 $\mathcal{L}$ 类)。^①

① 我们还要指出, $BPL \subseteq SC$, 其中 BPL 的定义在 6.1.4.1 节中给出, 这个结论将在 8.4 节得到证明 (见定理 8.23)。

通常可定义 $DT_I S_p(t, s)$ 是可由时间复杂度为 t 、空间复杂度为 s 的算法求解的判定问题类。注意: $DT_I S_p(t, s) \subseteq D_{\text{TIME}}(t) \cap D_{\text{SPACE}}(s)$ 且其中可能是严格的包含关系。我们指出, $DT_I S_p(\cdot, \cdot)$ 类提供了关于 $\mathcal{NP}$ 类的唯一已知(且非平凡)的绝对下界, 见参考文献[79]。还要注意的是一空折中(如参考文献[59]中 4.3 节)的下界也可用 $DT_I SP(\cdot, \cdot)$ 来描述。

5.1.4 电路求值

定理 3.1 断言, 存在多项式时间算法, 给定电路 $C: \{0, 1\}^n \rightarrow \{0, 1\}^m$ 及 n 比特串 x , 能返回 $C(x)$ 。对于有界扇入电路, 该算法的空间复杂度是电路深度(可能是电路规模的对数)的线性函数。这可由以下的 DFS-型算法得到。

算法(递归地)确定电路中每个门的取值, 首先确定第一条输入边的值, 再确定第二条输入边的取值。从而递归程序从电路的每个输出端开始运行, 运行中只需保存通向当前处理顶点的路径及其每个祖先结点的取值。注意: 在确定该路径时, 要为路径中每个结点标识出当前处理的是其第一条还是第二条输入边。如果处理的是结点的第二条输入边, 则还需保存其第一条输入边的取值。

从而, 上述算法对于输入 (C, x) 使用的临时存储空间为 $2d_c + O(\log|x| + \log|C(x)|)$, 其中 d_c 表示 C 的深度。其中第一项代表了算法的核心行为(即递归过程), 而其余项表示处理初始输入和最终输出的负载(即为 x 中的比特分配 C 中相应的输入端, 并扫描 C 的所有输出端)。

注意: 5.2.3 节及 5.3.2.2 节中将讲述电路复杂度与空间复杂度之间进一步的联系。

5.2 对数空间

虽然习题 5.3 断言“在对数空间之下还有生命”, 但对数空间仍被看作是支持有意义计算所需的最小空间。特别是, 对数空间只需设置一个辅助的计数器, 用以在输入中保留一个位置, 许多计算都似乎要求这一点。另一方面, 对数空间足以解决许多自然的计算问题, 在计算问题间构造归约, 以及严格地定义(布尔函数类间的)一致性。事实上, 对数空间运算的一个重要特征是它们是多项式时间运算的一个自然子类(见定理 5.3)。

5.2.1 L 类

主要讨论判定问题, 定义 $\mathcal{L}$ 类为可以利用对数空间复杂度算法求解的判定问题类, 即 $L = \bigcup_c D_{\text{SPACE}}(l_c)$, 其中 $l_c(n) \stackrel{\text{def}}{=} c \log_2 n$ 。注意: 由定理 5.3, $\mathcal{L} \subseteq \mathcal{P}$ 。已经暗示过许多自然的计算问题属于 $\mathcal{L}$ 类(见习题 5.6、习题 5.12 及 5.2.4 节)。另一方面, 普遍认为 $\mathcal{L} \neq \mathcal{P}$ 。

5.2.2 对数空间归约

另一类重要的对数空间计算是对数空间归约。根据 5.1.3.3 节中的细节讨论, 我们只考虑多到一映射的情形。类似于 Karp-归约的定义(定义 2.11), 称 f 是 S 到 S' 的对数空间(多到一)归约, 如果 f 可在对数空间内计算且对任意 x , $x \in S$ 当且仅当 $f(x) \in S'$ 。由引理 5.2(及定理 5.3)可知, 如果 S 可以对数空间归约到 $\mathcal{L}$ 中某个集合, 则 $S \in \mathcal{L}$ 。类似地, 可以定义对数空间 Levin-归约(定义 2.12)。这两种归约都具有传递性(见习题 5.11)。注意: 定理 5.3

可在这里应用并暗示着这些归约运行于多项式时间。因此,对数空间多到一归约是 Karp-归约的特殊情形。

注意到,所有已知的用于证明 NP-完全性结论的 Karp-归约实际上都是对数空间归约。对于 2.3.3 节(以及 2.3.2 节)中的归约易验证这一点。例如,考虑定理 2.21 证明中到 CSAT 的构造性归约:所构造的电路是“强一致性的”并且易在对数空间内构造(还可见于 5.2.3 节)。这个归约的退化形式可用于证明: $\mathcal{P}$ 类中每个问题都可以在对数空间内归约为对给定电路和给定输入求值的问题。注意:后者属于 $\mathcal{P}$ 类,从而可以说,该问题在对数空间归约下是 $\mathcal{P}$ -完全的。

定理 5.4 (电路求值的复杂性) 令 CEVL 表示形如 (C, α) 的对集合,其中 C 为布尔电路且 $C(\alpha)=1$, 则 CEVL 属于 $\mathcal{P}$ 且 $\mathcal{P}$ 中任意问题都可在对数空间内 Karp-归约为 CEVL。

证明概要: 回顾在定理 2.21 的证明中(以及定理 3.6 的证明中)可以观察到,图灵机的计算可以用(“强一致性的”)电路类来模仿。该证明将归约的输入(记作 x)硬连接到电路(记作 C_x)中,并令输入端对应于 NP-证据(记作 y)中的比特。在这里仍令 x 为电路的输入,但是要注意作为辅助输入的 NP-证据并不存在(或长度为 0)。因此,从 $S \in \mathcal{P}$ 到 CEVL 的归约将 $(S$ 的)实例 x 映射为一个对 $(C_{|x|}, x)$, 其中 $C_{|x|}$ 为模仿(对任意 $|x|$ 比特长输入)判定 S 成员机器的电路。为便于将来使用(5.2.3 节),我们强调当给定输入 $1^{|x|}$ 时, $C_{|x|}$ 可由对数空间机器构造。 $\square$

对数空间归约对于 $\mathcal{P}$ -完全性的影响

事实上,定理 5.4 暗示了 $\mathcal{L} \neq \mathcal{P}$ 当且仅当 $\text{CEVL} \notin \mathcal{L}$ 。可以证明其他一些自然问题也具有这个性质(即在对数空间归约下的 $\mathcal{P}$ -完全性,见参考文献[60])。

人们假设在证明关于其他类的完全性时使用的对数空间归约超出了 $\mathcal{L}$ 类的范围。当考虑的问题类包含于 $\mathcal{P}$ 类中时(如 $\mathcal{NL}$ 类,见 5.3.2 节),必须对归约能力做出这种限制。一般来说,称问题 Π 在对数空间归约下是 C -完全的,如果 Π 属于 C 且 C 中任意问题都可在对数空间内(多到一地)归约为 Π 。此时,如果 $\Pi \in \mathcal{L}$, 则 $C \in \mathcal{L}$ 。

与多项式时间归约相同,我们强调对数空间归约的用途应该不仅仅是证明完全问题。

5.2.3 对数空间一致性及更强的概念

定义 3.3 中引入了电路类 $(C_n)_{n \in \mathbb{N}}$ 一致性的基本概念,要求存在一个算法,对于输入 n ,输出关于电路 C_n 的描述,并且该算法所需时间是 C_n 规模的多项式。现在对定义 3.3 进行强化,称电路 $(C_n)_{n \in \mathbb{N}}$ 是对数空间一致的,如果存在一个算法,对于输入的 n 输出 C_n , 且算法使用的空间是 C_n 规模的对数。如以下定理 5.5 所暗示的(已经在前面的定理 5.4 中得到证明),任意一个多项式时间算法可以用对数空间一致的多项式规模(有限扇入)电路来模仿。另一方面,续 5.1.4 节,要注意具有有限扇入及对数深度的对数空间一致电路可以用具有对数空间复杂度的算法来模仿(即“对数空间一致的 $\mathcal{NC}^1$ ”属于 $\mathcal{L}$, 见习题 5.12)。

3.1.1 节提到,我们考虑过更强的一致性概念。具体而言,类似于附录 E.2.1.2 中的讨论,称电路类 $(C_n)_{n \in \mathbb{N}}$ 有强化的明确构造,如果存在一个运行于多项式时间和线性空间的算法,使得对于输入的 n 和 v , 算法返回顶点 v 在电路 C_n 中的标号以及其子结点列表(或指出 v 不是 C_n 中的顶点)。注意:如果 $(C_n)_{n \in \mathbb{N}}$ 具有强化的明确构造,则它是对数空间一致的,因为用于描

述 C_n 中顶点的串长度是 C_n 规模的对数。定理 5.4 实际上证明了如下结论。

定理 5.5 (模仿 $\mathcal{P}$ 的强一致性电路) 对任意多项式时间算法 A , 存在一个具有强化明确构造的多项式规模电路类 $(C_n)_{n \in \mathbb{N}}$, 使得对任意 x , 有 $C_{|x|}(x) = A(x)$ 。

证明概要: 已经指出, (定理 5.4 证明中用到的) 电路 $(C_{|x|})_{|x|}$ 是强一致的。特别地, 电路所体现的 (有向) 图中包含了常数规模的分图, 排列成一个阵列并只与相邻的分图连通 (见定理 2.21 的证明)。□

5.2.4 无向连通性

对图进行探索 (如为判定图的连通性) 是图论中最重要而常见的基本计算问题之一。标准的图探索算法 (如 BFS 和 DFS) 需要的临时存储空间是图中顶点个数的多项式。相反, 本节提出的算法使用的临时存储空间只是顶点个数的对数。除了用于说明对数空间计算的能力之外, 在复杂的对数空间算法的设计中, 本节的算法 (或其实现) 代表着一类设计风格。

直观上, “探索一个图” 可以理解为判定一个给定图是否为连通的。^① 对于这个自然的计算问题, 除了讨论其本质意义之外, 我们还指出, 它在计算上等价于 (在对数空间归约下) 大量的其他计算问题 (见习题 5.16)。注意: 某些相关的计算问题实际上似乎更难, 例如, 判定有向图的有向连通性, 这个问题带有 $\mathcal{NL}$ 类的性质 (见 5.3.2 节)。在这种情况下, 我们强调一个事实, 即这里考虑的计算问题都只针对无向图, 并称其为图的无向连通性。

定理 5.6 判定图的无向连通性 (UCONN) 属于 $\mathcal{L}$ 。

判定算法基于这样一个事实, 即当图中包含常数次扩张集时, UCONN 是容易的。^② 特别地, 如果图的度数为常数, 且具有对数直径, 则可在对数空间内进行探索 (用于确定一条从固定起点开始的新路径)。^③

当然, 输入的图中不一定包含常数次扩张集。因此, 证明的主要思想是将输入的图转化为一个满足上述条件的图, 同时保持图中连通分量个数不变。另外, 关键是这种转化要在对数空间内实现。本节的剩余部分将详细描述这种转化。我们首先提出一种基本方法, 然后介绍对于非平凡情形的实现细节。

教学注释: 建议将定理 5.6 的证明 (即本节剩余部分) 作为进一步的学习内容。因为其中使用了许多超出复杂性理论范围的技术。

首先要注意, 将输入的图 $G_0 = (V_0, E_0)$ 转化为常数度数的图 G_1 , 且保持连通分量个数不变, 这是很容易的。具体而言, (G_0 中) 每个具有度数 $d(v)$ 的顶点 $v \in V$ 可用 (G_1 中) 含 $d(v)$ 个顶点的圈 C_v 来代替, 而每条边 $\{u, v\} \in E_0$ 用一个一端在圈 C_v 中, 另一端在圈 C_u 中的边代替, 从而使得 G_1 中每个顶点的度数都为 3 (即有两个圈内边, 一个圈间边)。这种变换可以在对数空间内实现, 从而 (由引理 5.2) 可以假设输入的图度数为 3。

我们的目标是要将该图转化为扩张集, 并保持连通分量个数不变。事实上, 在描述这种

① 基本术语可见附录 G.1。

② 读者可能会认为扩张仅仅是对数直径的图。在随后更深入的证明阶段, 必须假设读者熟悉扩张的定义, 以及构造扩张的具体技术。相关材料包含于附录 E.2 中。

③ 事实上, 这与 5.1.4 节的电路求值算法很相似, 其中电路深度对应于图的直径, 有限扇入对应于常数度数。进一步的细节可见习题 5.13。

转化时,可以假设图是连通的,并特别说明如果不连通,则转化过程在各个连通分量中独立进行。

几个技术细节

设常数 $d > 2$ 是为满足某种附加条件而确定的,可以假设输入的图实际上是 d^2 -正则的(虽然不一定是简单图)。进一步地,假设该图不是二部图。为使两个假设合理化,可以为如上构造的 3-正则图中的每个顶点增加一个 $d^2 - 3$ 的自环。

预备知识

显然,扩张图在上述转化中起重要作用。读者可以参考附录 E.2 中的相关概念。特别地,假设读者熟悉扩张的代数定义(见附录 E.2.1.1)。进一步地,上述转化还大量使用了附录 E.2.2.2 中的 Zig-Zag 乘积,而以下假设读者熟悉该乘积的定义。

5.2.4.1 基本方法

回想我们的目标是将图 G_1 转化为一个扩张。转化的过程是逐步实现的,其中含对数次迭代,每次迭代使扩张参数加倍,而图的规模也呈常数因子地增加并保持度数界限。这里所说的(扩张)参数是指相对特征值缝隙(见附录 E.2.1.1)。^①其值为常数时表明该图是一个扩张图。一开始,这个参数的下限是 $\Omega(n^{-2})$,其中 n 为图的规模。由于每次迭代中参数会加倍,经过对数次迭代之后,该参数的下限将为一个常数(从而当前的图为一个扩张)。

上述逐步转化过程中的一个难点是,每次迭代中都会发生转化。假定转化过程使扩张因子加倍,同时保持图的度数不变并将顶点个数增加一个常数因子。转化的过程结合了(标准的)图的求幂操作以及附录 E.2.2.2 中的 Zig-Zag 乘积。具体地,对于足够大的整数 d 和 c ,我们从一个 d^2 -正则图 $G_1 = (V_1, E_1)$ 开始,经过对数次的迭代,对于 $i = 1, 2, \dots, t-1$,令 $G_{i+1} = G_i^c \text{ ZZ } G$,其中 G 为一个固定的含 d^{2c} 个顶点的 d -正则图。即在每次迭代中,将当前图(即 G_i)提升到 c 次幂,并将得到的图(d^{2c} -正则的)与固定的(含 d^{2c} 个顶点的)图 G 求 Zig-Zag 乘积。因此, G_{i+1} 为含 $d^{i \cdot 2c} \cdot |V_1|$ 个顶点的 d^2 -正则图,其中的不变量由 Zig-Zag 乘积的定义来保持(即 d^{2c} -正则图 $G' = (V', E')$ 与 d -正则图 G (含 d^{2c} 个顶点)的 Zig-Zag 乘积是一个含 $d^{2c} \cdot |V'|$ 个顶点的 d^2 -正则图)。

可用式 (E.11) 来分析扩张参数(即相对特征值缝隙),记作 $\delta(\cdot) \stackrel{\text{def}}{=} 1 - \bar{\lambda}(\cdot)$ 的增长情况。式 (E.11) 蕴含了,如果 $\bar{\lambda}(G) < 1/2$,则 $1 - \bar{\lambda}(G' \text{ ZZ } G) > (1 - \bar{\lambda}(G'))/3$ 。因此,选择的图 G 必须满足 $\bar{\lambda}(G) < 1/2$,这就要求常数 d 足够大。从而有

$$\delta(G_{i+1}) = 1 - \bar{\lambda}(G' \text{ ZZ } G) > \frac{1 - \bar{\lambda}(G_i)^c}{3} = \frac{1 - \bar{\lambda}(G_i)^c}{3}$$

然而,对于足够大的常数 $c > 0$,有 $1 - \bar{\lambda}(G_i)^c > \min(6 \cdot (1 - \bar{\lambda}(G_i)), 1/2)$ 。^②于是,有

① 一个正则图的特征值缝隙是图的度数(即图的最大特征值)与所有其他特征值的绝对值之差。相对特征值界限,记作 $\bar{\lambda}$,是特征值界限除以图的度数的结果。

② 考虑以下两种情况:当 $\bar{\lambda}(G_i) < (1 - (1/c))$ 时,证明 $1 - \bar{\lambda}(G_i)^c > 1/2$,否则,令 $\varepsilon = 1 - \bar{\lambda}(G_i)$,并利用 $\varepsilon \leq 1/c$ 来证明 $1 - \bar{\lambda}(G_i)^c > c\varepsilon/2$ 。

$\delta(G_{i+1}) > \min(2\delta(G_i), 1/6)$ 。因此, 令 $t = O(\log|V_i|)$ 并利用 $\delta(G_i) = 1 - \bar{\lambda}(G_i) = \Omega(|V_i|^{-2})$, 可得到 $\delta(G_i) > 1/6$ 。

毫无疑问, 一个非常重要的“细节”是要利用对数空间算法将 G_i 转化为 G_t 。事实上, G_i 到 G_{i+1} 的转化可在对数空间内实现 (见习题 5.14), 但是需要构造对数个这样的转化。不幸的是, 我们无法提供空间受限算法的标准组合引理中所要求的负载。^①然而, 如果更仔细地观察由 G_i 到 G_{i+1} 的转化过程, 会注意到它具有很强的结构, 从而在某种意义上可在常数空间内实现, 并支持更强的组合结论, 使每次迭代只需要常数的存储空间。最终得到的实现 (从 G_i 到 G_t 的迭代转化) 及其形式化定义将在 5.2.4.2 节中给出 (参考文献[190]中提出了另一种实现, 可通过对组合的拆分而得到)。

5.2.4.2 实际实现

为了高效地实现 5.2.4.1 节中的迭代转化过程, 观察到我们不需要明确地构造出各种图, 而仅提供对其“预言访问”即可。这一论断对于中间图来说很关键, 即当给定 G_i 作为输入时, 不是构造 G_{i+1} , 而是说明在给定了向 G_i 的预言访问时, 如何提供向 G_{i+1} 的预言访问 (即回答关于 G_i 的“相邻询问”)。这意味着, 可将 G_i 和 G_{i+1} (而非其依附列表) 看作是 (要求值的) 函数, 而不是 (要打印的) 串, 并说明如何将 G_{i+1} 中寻找邻居简化为在 G_i 中寻找邻居。

讨论

注意: 这里所指的预言机是有限预言, 表示一个有限的可变对象 (这又是某些计算问题的一个实例)。这种机器通过访问一个基本的对象来提供向一个更复杂对象的访问。具体地, 这种机器得到一个输入, 这是关于该复杂对象 (即机器要模仿的对象) 的“询问”, 并产生一个输出 (这是对该询问的应答)。类似地, 这些机器发出关于另一个对象的询问 (即预言所代表的对象), 并得到相应的应答。^②

与 5.1.3.3 节相似, 询问通过一个特殊的只写设备发出, 应答则从相应的只读设备中读出, 这些设备占用的空间并不计入空间复杂度。根据这些习惯, 我们断言, 在 d^2 -正则图 G_{i+1} 中的邻居可用一个常数空间预言机来计算, 该机器可以访问 d^2 -正则图 G_i 。也就是说, 令 $g_i: V_i \times [d^2] \rightarrow V_i \times [d^2]$ (或 $g_{i+1}: V_{i+1} \times [d^2] \rightarrow V_{i+1} \times [d^2]$) 表示 G_i (或 G_{i+1}) 的边旋转函数 (Edge-rotation Function) ^③。

断言 5.7 存在一个常数空间预言机, 当给定向 g_i 的预言访问时, 可以计算 g_{i+1} , 其中机器的状态计入空间复杂度中。

证明概要: 首先证明, G_{i+1} 定义中的两种基本操作 (即求幂和与给定图的 Zig-Zag 乘积) 可在常数空间内实现。

可对任意的对计算 G_i^2 (即图 G_i 的平方) 的边旋转函数, 方法是计算两次 G_i 的边旋转函

① 我们无法提供简单组合 (引理 5.1), 因为其所需负载是中间结果规模的线性函数。在仿真组合 (引理 5.2) 中, 需要将组合算法的空间复杂度相加 (更不用说还要加另一个对数项), 这将导致空间复杂度上限达到对数平方级。

② 事实上, 当前环境 (其中的预言机表示有限的可变对象, 这反过来又是某个计算问题的一个实例) 不同于标准环境, 其中的预言表示一个固定的计算问题。然而, 这两种类型的预言机的运行机制 (和/或操作) 是相同的: 都是得到一个输入 (在这里是一个关于可变对象的“询问”, 而非关于某个给定计算问题实例的询问), 并产生一个输出 (这里是对询问的应答, 而非对于给定实例的“解”)。类似地, 这些机器发出询问 (这里的询问是关于另一个可变对象的, 而不是关于另一个给定计算问题实例), 并得到相应的应答。

③ 图的边旋转函数将对 (v, j) 映射为对 (u, k) , 如果顶点 u 是顶点 v 的第 j 个邻居, 且 v 为 u 的第 k 个邻居 (见附录 E.2.2.2)。

数, 并利用常数空间完成。具体而言, 给定 $v \in V_i$ 及 $j_1, j_2 \in [d^2]$, 计算 $g_i(g_i(v, j_1), j_2)$, 这是 G_i^2 中 $(v, \langle j_1, j_2 \rangle)$ 的边旋转, 计算方法如下: 首先, 发出询问 (v, j_1) , 得到 (v, j_1) 的边旋转, 记作 (u, k_1) 。下一步, 发出询问 (u, j_2) , 得到 (w, k_2) , 最后输出 $(w, \langle k_2, k_1 \rangle)$ 。我们强调, 临时空间只用于记录 k_1 , 而 u 的值直接从预言应答设备复制到预言询问设备中。记录下上述行为中各个阶段所需的常数个状态之后, 我们得到结论, 求图的平方可在常数空间内完成。可将这个结论扩展为图的任意次幂。

现在考虑 (任意一个正则图 G' 与固定图 G 的) Zig-Zag 乘积, 注意到, 相应的边旋转函数可以在常数空间内计算 (在给定向 G' 的边旋转函数的预言访问时)。这可直接由式 (E.9) 得到。注意到, 后者需要调用一次 G' 的边旋转函数, 并只依赖于常数规模的图 G 对该函数进行两个简单的修改 (使其只影响相关串中常数个比特)。另外, 由于可将顶点名称从输入复制到预言询问设备 (或由预言应答设备复制到输出), 我们得到结论, 上述行为可在常数空间内完成。

可将结论扩展到对上述类型操作的常数次组合 (即求图的平方以及与固定图的 Zig-Zag 乘积)。□

递归组合

利用断言 5.7, 我们希望得到一个 $O(t)$ -空间预言机, 通过发出向 g_1 的预言调用来计算 g_t , 其中 $t = O(\log |V_i|)$ 。从这样的预言机出发可以得到一个由 G_1 到 G_t 的对数空间转换 (通过对所有可能的值计算 g_t)。我们希望组合引理最好是充分的, 将其应用于断言 5.7, 可以得到期望中的 $O(t)$ -空间预言机 (从而把 g_t 的求值简化为对 g_1 的求值)。实际情况正是如此, 除了尚未得到充分的组合引理之外 (下面将解决这个问题)。

首先要注意, 使用简单组合 (如引理 5.1) 时, 每次组合都会增加一个额外的负载 $O(\log |V_i|)$ 。但是我们无法为每次组合提供多于常数的负载。使用仿真组合时 (如引理 5.2), 将为每次组合引入乘法的负载, 而这显然做不到。以下介绍的组合方法是简单组合的变形, 在递归调用中很有用。其基本思想是不采用为递归的每一层分配独立的输入/输出和询问设备的传统做法, 而是将所有设备合成为一个 (“全局性的”) 设备, 它可用于递归的所有层次。就是说, 不是根据 “结构化程序设计” 方法, 利用本地分配的空间将信息传向子程序, 而是借鉴 “不好的编程思想”, 利用全局变量来传递信息 (照例, 这一概念是用多带图灵机模型定义的, 但是也可以使用其他合理的计算模型)。

定义 5.8 (全局带预言机) 全局带预言机是一种预言机 (见定义 1.11), 其中的输入、输出和预言带都用一个全局带来代替。另外, 机器中有常数个工作带, 称为本地带。机器从全局带上得到输入, 将每个询问写入这个带中, 并从该带上得到相应的应答, 最后把最终输出也写到这个带上 (我们强调, 调用预言 f 的结果使全局带的内容由 q 变为 $f(q)$)。^①在讨论这个机器的空间复杂度时, 需要分开考虑全局带和本地带。

显然, 任意一个普通的预言机都可转化为等价的全局带预言机。最终得到一个使用长度不超过 $n + l + m$ 的全局带的机器, 其中 n 表示输入长度, l 表示最长询问或应答的长度, m

^① 这意味着全局带的前一个内容 (即询问 q) 丢失了 (即被替换为应答 $f(q)$)。因此, 如果想要保存前面的内容, 就必须将其复制到本地带上。还要强调的是, 根据标准的预言调用习惯, 预言应答之后的头位置位于全局带的最左边单元。

为输出长度。然而,将这3个不同的带合成为一个全局带似乎需要为每个带保持独立的指针,这就意味着必须在本地带上存储3个相应的计数器(除了存储原始工作带以外)。因此,最终得到的机器使用的全局带长度为 $w + \log_2 n + \log_2 l + \log_2 m$, 其中 w 表示原始机器的空间复杂度,其余的对数项(是全局带长度的对数)表示上述3个计数器。

幸运的是,当原始预言机被描述为迭代的变换序列时(即输入变成第一个询问,而第 i 个应答变成第 $i+1$ 个询问或输出,同时保存工作带上的辅助信息),则可以避免使用上述三个计数器。事实上,断言 5.7 的证明中构造的机器就具有这种形式,从而可以用一个全局带预言机来实现,其中全局带长度不超过输入和具有固定长度的本地带长度(本地带的长度并非全局带长度的对数)。

断言 5.9 (断言 5.7 的再次描述) 存在一个全局带预言机,在给定向 g_i 的预言访问时,利用长为 $\log_2(d^2 \cdot |V_{i+1}|)$ 的全局带和长度固定的本地带,可以计算 g_{i+1} 。

证明概要: 根据断言 5.7 的证明,只需指出恰好使用两个带即可。例如,回顾对 G_i 的平方在点 $(v, \langle j_1, j_2 \rangle)$ 计算边旋转函数时,可以先求原始图在 (v, j_1) 的边旋转函数,再求 (u, j_2) 的,其中 $(u, k_1) = g_i(v, j_1)$ 。这意味着全局带机器首先从全局带上读入 $(v, \langle j_1, j_2 \rangle)$, 再用将其用询问 (v, j_1) 代替,并将 j_2 保存在本地带上。因此,机器仅仅删除全局带上的常数个比特(并保持其前缀不变)。调用预言之后,机器从全局带(目前保存的是 (u, k_1))复制下 k_1 , 保存于本地带上,并将 j_2 从本地带复制到全局带上(从而使其包含 (u, j_2))。在第二次调用预言之后,全局带上包含了 $(w, k_2) = g_i(u, j_2)$, 且机器仅仅将其修改成 $(w, \langle k_2, k_1 \rangle)$, 这就是需要的输出。

类似地,注意:在 $(\langle u, i \rangle, \langle \alpha, \beta \rangle)$ 计算图 G' 与给定图 G 的 Zig-Zag 乘积的边旋转函数时,可对 $(u, E_\alpha(i))$ 询问 G' , 并输出 $(\langle v, E_\beta(j') \rangle, \langle \beta, \alpha \rangle)$, 其中 (v, j') 表示预言应答(见式 (E.9))。这意味着全局带机器首先从全局带向本地带复制 α, β , 将全局带内容由 $(\langle u, i \rangle, \langle \alpha, \beta \rangle)$ 修改为 $(u, E_\alpha(i))$, 再在预言调用之后进行类似转化。□

全局带预言机的构成

断言 5.9 的证明中非显式地使用了全局带预言机上计算的串行组合。^①一般而言,在对这些计算进行串行组合时,全局带(或本地带)的长度是所有组合运算的最大长度,也就是说,目前的定义提供了简单串行组合的紧致界限(不同于引理 5.1)。另外,全局带预言机在递归组合意义下是有用的,如引理 5.10 所示(该引理严格依赖于这个模型)。关键在于递归组合中的所有层次都将重复使用相同的全局存储空间,只有本地存储在增加。因此,有如下的组合引理。

引理 5.10 (全局带模型下的递归组合) 假设存在一个全局带预言机,对任意 $i=1,2,\dots,t-1$, 通过向 f_i 发出预言询问并利用长为 L 的全局带和长为 l_i 的本地带来计算 f_{i+1} , 其中本地带还用于保存机器状态。则利用标准的预言机并向 f_1 发出预言调用,可在空间 $L + \sum_{i=1}^{t-1} (l_i + \log_2 l_i)$ 内计算 f_t 。

在应用这个引理时,可令 $f_i = g_i$, $t = O(\log |V_1|) = O(\log |V_t|)$, 并设置上限 $L = \log_2(d^2 \cdot |V_t|)$ 以及 $l_i = O(1)$ (如断言 5.9 中所保证的)。事实上,在这个应用中, L 等于输入 $f_t = g_t$ 的长度。

① 断言 5.7 的证明中描述了一个类似的组合,但是断言 5.9 表明这种计算有更强的特征。

证明概要：通过在每层递归中对全局带和本地带的模拟过程分配空间来计算 f_i 。在模拟递归运算时，强调所有的递归都使用同一个全局带（用于发出询问及接收应答）。回顾在实际递归的每一层中，只要在将控制返回给调用机器时，全局带上有正确应答，便可以随意使用同一个全局带。因此，模拟过程可以是相同的，利用分配给全局带的空间以及分配给这一层中本地带的空间，可以模拟每一层。在模拟中，还要将其他层的位置保存于相应的本地带上，但是这个操作所需空间（即 $\sum_{i=1}^{t-1} \log_2 l_i$ ）显然小于各个本地带的长度（即 $\sum_{i=1}^{t-1} l_i$ ）。□

结论

结合断言 5.9 及引理 5.10，可以得到结论，对于 $g_{O(\log|V_1|)}$ 的求值可在空间 $O(\log|V_1|)$ 内简化为对 g_1 的求值，即 $g_{O(\log|V_1|)}$ 可以用一个调用 g_1 的标准预言机计算，使用的空间为 $O(\log|V_1|)$ 。回顾 G_1 可在对数空间内构造（基于输入图 G_0 ），我们推断 $G' = G_{O(\log|V_1|)}$ 也可在对数空间内构造。由定理 5.6 可知， G' （具有常数度数及对数直径）的连通性可在对数空间内判定（见习题 5.13）。利用类似方法可以验证给定的一对顶点在输入图中是否连通（见习题 5.15）。另外，在相同空间复杂度内可以找到一条相应的路径（见习题 5.17）。

5.3 非确定的空间复杂性

空间复杂性与时间复杂性的差别在非确定计算中尤为显著。一个现象是，非确定空间复杂度的两种定义间存在巨大差异（见 5.3.1 节），这完全不同于在时间复杂性中类似的两个定义是等价的。我们还要对比有关（标准模型下）非确定的空间受限计算的结论（见 5.3.2 节）以及时间复杂性中的类似结论，一个很好的例子是“补问题”（见 5.3.2.3 节）。

5.3.1 两个模型

前面在定义非确定的时间受限计算时，使用了两个等价模型。在离线模型中（将 NP 定义为一个证明系统（见定义 2.5）），非确定性被理解为要从现有的（“非确定的”）辅助输入中选择。因此，在这种模型下，非确定性是指超出了机器本身的选择（即执行“离线”选择）。相反，在在线模型中（使用 NP 的传统定义（见定义 2.7）），非确定性被理解为要参考机器本身做出的非确定选择。在时间复杂性中，这些模型是等价的，因为记录在线选择（几乎）无需任何代价（见定理 2.8 的证明）。然而，在空间复杂性中，这种记录需要代价。

再仔细研究一下离线模型与在线模型的关系。一个突出问题是，用在线选择来模拟离线选择的代价是多少，以及相反情况。在此强调一个事实，在离线模型下，非确定的选择被“自由地”记录于一个辅助输入设备上，而这个自由记录对于在线模型是不可能的。在线模型可由离线模型自由地模拟，这个事实显然成立，即对任意自然的复杂性概念都成立，因为在线的非确定性选择可以利用（离线的）非确定输入中的连续比特来模拟（而不会严重影响任何自然的复杂性度量）。相反，用在线模型高效地模拟离线模型则要依赖于（在在线模型下）高效地保存非确定选择的记录的能力，这就抵消了离线非确定输入（在离线模型中是自由记录的）的优势。这个高效的模拟在时间复杂性下是可能的，因为此时机器可以保存一个非确定选择（在线进行）的序列，并恢复其中的比特，而不明显影响运行时间（即几乎是“免费的”）。这种利用在线选择来简单模拟离线选择的方法，在空间受限计算中并不是“免费的”，因为（在

在线模型中)记录(即保存)下的每个在线选择都会使空间复杂度增加。另外,典型地,非确定选择的数量通常远远大于空间上限,因此,简单模拟在空间复杂度中无法实现(因为在空间复杂性意义下代价太大)。

让我们扼要重述一遍两个模型,并考虑在空间复杂度中两者的关系。在标准模型中,称为在线模型,机器“在线地”做出非确定的选择(如定义 2.7)^①。因此,如果机器在后续计算中需要参考这样一个非确定的选择,则必须将其保存于存储设备中(并付出空间代价)。相反,在所谓的离线模型中,非确定选择作为一个特殊的非确定输入由外部提供。这个非确定的输入来自于一个特殊的只读设备(或带),可以像主要输入一样从两个方向扫描。

将可由空间复杂度为 s 的在线(或离线)非确定机器接受的集合记作 $N_{\text{SPACEon-line}}(s)$ (或 $N_{\text{SPACEoff-line}}(s)$)。我们强调,如定义 2.7 所述,被非确定机器 M 接受的集合由串 x 构成,满足 M 对于输入 x 存在一种可接受的计算(在在线模型下,这里所说的存在性是指机器自己做出的非确定选择,而在离线模型下,是指相应的非确定输入)。

这两个类之间的关系并不明显。事实上, $N_{\text{SPACEon-line}}(s) \subseteq N_{\text{SPACEoff-line}}(s)$, 但是(一般情况下),这种包含关系在反方向并不成立。事实上,当 s 至少为对数时,不仅仅有 $N_{\text{SPACEon-line}}(s) \neq N_{\text{SPACEoff-line}}(s)$, 而且 $N_{\text{SPACEon-line}}(s) \subseteq N_{\text{SPACEoff-line}}(s')$, 其中 $s' = O(\log s(n)) = o(s(n))$ 。进一步地,当 s 至少为线性时,有 $N_{\text{SPACEon-line}}(s) = N_{\text{SPACEoff-line}}(\Theta(\log s))$, 见习题 5.18。

在进一步讨论之前,首先要说明,本节其余部分为什么将重点放在在线模型。事实上,离线模型更适合 $\mathcal{NP}$ 类(如 2.1.2 节所述),但是在空间复杂度意义下研究非确定性时,在线模型似乎更充分。原因之一是一个离线的、非确定的输入可以用来对计算编码(见习题 5.18),并且在某种意义上允许对于计算所需的空间复杂度进行“欺骗”。这体现在离线模型可以模拟在线模型,而使用的空间是在线模型使用空间的对数。一个相关现象是只知道 $N_{\text{SPACEoff-line}}(s)$ 包含于 $D_{\text{TIME}}(2^{2^s})$ 中,而 $N_{\text{SPACEon-line}}(s) \subseteq D_{\text{TIME}}(2^s)$ 。这个事实引发了对 $\mathcal{NL} = N_{\text{SPACEon-line}}(\log)$ 的研究,它是 $\mathcal{P}$ 类的(自然)子类。事实上,有关 $\mathcal{NL}$ 的各种结论回过头来也印证了其研究的合理性。

根据上述讨论,我们采用标准习惯,令 $N_{\text{SPACE}}(s) = N_{\text{SPACEon-line}}(s)$ 。我们主要讨论 $\mathcal{NL} = N_{\text{SPACE}}(\log)$ 。5.3.2 节对这一类进行研究之后,5.3.3 节又回到“模型问题”。

5.3.2 NL 及有向连通性

本节讨论 $\mathcal{NL}$ 类,我们将其视为非确定的 $\mathcal{L}$ 类。具体地, $\mathcal{NL} = \bigcup_c N_{\text{SPACE}}(l_c)$, 其中 $l_c(n) = c \log_2 n$ (建议读者参考 5.3.1 节讨论的关于 $N_{\text{SPACE}} = N_{\text{SPACEon-line}}$ 的定义)。

首先要注意,定理 5.3 的证明易扩展到(在线的)非确定情形。原因在于,由确定模型转换为当前模型不会影响即时配置(定理 5.3 的证明中给出了定义)的数量,而这个数量正是时间复杂性的上限,因此 $\mathcal{NL} \subseteq \mathcal{P}$ 。

以下问题称为有向连通性(st-CONN),体现了非确定对数空间计算的本质(并且,特别地,它在对数空间归约下是 $\mathcal{NL}$ 完全的)。st-CONN 的输入包括一个有向图 $G = (V, E)$, 及一对顶点 (s, t) , 其任务是确定(图 G 中)是否存在一个从 s 到 t 的有向路径。^②事实上,对于 $\mathcal{NL}$ 的研究常常要借助于 st-CONN 问题。例如,注意到由 st-CONN 属于 $\mathcal{P}$ 这一事实(以及

① 另一种等价定义中,考虑由一个特殊的只读带读取非确定输入的机器,这个带只能从一个方向读。这与离线模型不一致,后者的非确定输入来自一个可自由扫描的只读带。

② 见附录 G.1 中的图论基本术语。注意:这里(以及下面的内容中) s 表示起点, t 表示终点。

$\mathcal{NL}$ 可在对数空间内归约到 st-CONN), 易得到 $\mathcal{NL} \subseteq \mathcal{P}$ 。

5.3.2.1 完全性及其他

显然, st-CONN 属于 $\mathcal{NL}$ (见习题 5.19)。 st-CONN 在对数空间归约下的 $\mathcal{NL}$ -完全性可利用以下事实来证明, 即任意非确定的空间受限机器可生成一个有向图, 其中的顶点对应于机器的可能配置, 而边代表计算的“连续”关系。特别地, 对于对数空间计算, 该图是多项式规模的, 但是, 一般来说, 相应的图非常明确 (在自然意义上, 见习题 5.21)。

定理 5.11 $\mathcal{NL}$ 中任何问题都可在对数空间内归约到 st-CONN (通过多到一映射)。

证明概要: 给定一个非确定的 (在线) 机器 M , 以及一个输入 x , 我们考虑如下的有向图 $G_x = (V_x, E_x)$ 。 V_x 中的顶点是 $M(x)$ 的可能配置, 其中每个配置包括工作带内容 (以及机器的有限状态), 机器在工作带上的位置, 以及机器在输入带上的位置。图中的有向边表示一次计算行为。我们强调, 该行为取决于机器 M 以及 x 中的 (单个) 比特, 而 x 的位置由初始配置规定 (即对应于可能的边起点的配置)^①。注意: (对于固定的机器 M) 给定 x , 图 G_x 可在对数空间内构造 (扫描所有的顶点对, 并只输出对应于有效边的顶点对 (这又可以在常数空间内验证))。

由定义可知, 图 G_x 表示 M 对于输入 x 的可能计算, 特别地, M 对于 x 存在一种可接受的计算当且仅当 G_x 中存在一条有向路径, 起点为 s , 对应于初始配置, 而终点是 t , 对应于一种规范的可接受配置。因此, $x \in S$ 当且仅当 (G_x, s, t) 是 st-CONN 的一个“是”-实例。 $\square$

思考

我们认为, 定理 5.11 的证明 (还可见于习题 5.21) 表明, st-CONN 体现了非确定性空间受限计算的本质, 这个说法完全合理。注意这种 (直观的且非正式的) 表述超出了称 st-CONN 在对数空间归约下是 $\mathcal{NL}$ -完全的。

注意: 无向连通性 (见定理 5.6 及习题 5.15) 及有向连通性 (见定理 5.11 及习题 5.23) 在空间复杂度上是存在差异的。这里有必要提醒读者, 在无向 (或有向) 图中判定相对较短的路径 (而非任意路径) 存在与否, 这一问题在对数空间归约下也是 $\mathcal{NL}$ -完全的, 见习题 5.24。

st-CONN 的搜索版本

我们指出, 对应于 st-CONN 的搜索问题可在对数空间内归约到 $\mathcal{NL}$ (通过 Cook-归约), 见习题 5.20。还要注意, 任意一种可被对数空间非确定机器接受的计算, 可以通过求有向图中的有向路径来求得; 事实上, 这是对命题“ st-CONN 体现了非确定对数空间计算”的一种简单陈述。

5.3.2.2 NSPACE 到 DSPACE 的关联

回顾在时间复杂性中, 唯一已知的非确定计算到确定计算的转化会导致复杂度的指数增长。相反, 空间复杂度允许这种转化在复杂度上呈多项式增长。

定理 5.12 (非确定空间与确定空间) 对任意限定空间构造的至少为对数的函数 $s: \mathbb{N} \rightarrow \mathbb{N}$, 有 $N_{\text{SPACE}}(s) \subseteq D_{\text{SPACE}}(O(s^2))$ 。

特别地, 非确定的多项式空间包含于确定的多项式空间中 (且非确定的对数空间包含于确定的对数空间中)。

证明概要: 我们只讨论 $\mathcal{NL}$ 类的特殊情形, 得到的结果易推广到一般情形。或者, 由特

^① 因此, 实际的输入 x 只影响 G_x 的边集 (而顶点集只被 $|x|$ 所影响)。通过将输入带上的 x 中的 (单个) 比特集成到机器的配置中, 也可以得到一种相关的构造。在这种情况下, x 本身影响 V_x (但不影响 E_x , 除非 $E_x \subseteq V_x \times V_x$)。

殊情况下的结论出发,利用一个适当的上行转换引理(Upward-translation Lemma)(见参考文献[119]中12.5节),可得到一般情形下的结论。在特殊情形下,问题可归结为构造一个具有对数平方空间复杂度的算法,能够判定图的有向连通性。

基本思路是:将检查图 G 中 u 到 v 之间是否存在长度不超过 $2l$ 的路径这一问题(在对数空间内)归约为检查是否存在一个中间顶点 w ,满足存在一条长度不超过 l 、从 u 到 w 的路径,以及长度不超过 l 、从 w 到 v 的路径。也就是说,如果存在长度不超过 l 、从 u 到 v 的路径,则令 $\phi_G(u,v,l) \stackrel{\text{def}}{=} 1$,否则,令 $\phi_G(u,v,l) \stackrel{\text{def}}{=} 0$ 。从而,通过扫描 G 中所有的点 w ,可以计算出 $\phi_G(u,v,2l)$,并对每个 w ,验证是否 $\phi_G(u,w,l)=1$ 和 $\phi_G(w,v,l)=1$ 同时成立。^①因此,我们可以用一个对数空间算法计算 $\phi_G(u,v,2l)$,该算法调用预言 $\phi_G(\cdot,\cdot,l)$,这又可以用同样方式递归地求出。注意:原始的计算问题(即st-CONN)可以理解为对于给定的有向图 $G=(V,E)$ 和一对给定顶点 (s,t) ,计算 $\phi_G(s,t,|V|)$ (或 $\phi_G(s,t,2^{\lceil \log_2 |V| \rceil})$)。因此,如果我们应用充分的组合结论,则上述的递归程序服从定理中的断言。下面采用不同的技术来直接分析该递归程序。

回顾给定一个有向图 $G=(V,E)$ 及一对顶点 (s,t) ,可以只计算 $\phi_G(s,t,2^{\lceil \log_2 |V| \rceil})$,方法是扫描 G 中所有顶点并调用计算 $\phi_G(u,v,2l)$ 的递归程序,并对每个顶点 w ,计算 $\phi_G(u,w,l)$ 和 $\phi_G(w,v,l)$ 。巧妙之处在于,所有这些计算都可以重复使用相同的空间,从而我们只需要保存一个额外比特,代表着所有以前的计算结果即可。当且仅当对某个 w ,有 $\phi_G(u,w,l)=\phi_G(w,v,l)=1$ 时,返回的值为1(图5.2)。当然,易在对数空间内确定 $\phi_G(u,v,1)$ 的值。

考虑对上述程序(图5.2)的一种实现,在递归的每一层,在整个存储空间中指定一部分用于保存本地变量(即 w 和 σ)。每层递归所需空间为 $\log_2 |V|$ (用于存储 w 的当前值),而递归的层数为 $\log_2 |V|$ 。我们强调在计算 $\phi_G(u,v,2l)$ 时,发出了许多递归调用,但是所有调用重复使用同一个工作空间(即分配给当前层的部分)。也就是说,在计算 $\phi_G(u,v,l)$ 时,重复使用了计算 $\phi_G(u,w',l)$ 的空间, w' 为前面某次递归中的点,而在计算 $\phi_G(w,v,l)$ 时,也重复使用了同一空间。因此,算法的空间复杂度仅为所有递归层使用的空间之和。因此,st-CONN具有对数平方(确定的)的空间复杂度,而 $\mathcal{NL} \subseteq D_{\text{SPACE}}(O(\log^2))$ 中所有其他问题也是如此(因为st-CONN实际上代表了任何一种 $\mathcal{NL}$ 计算,或者因为 $\mathcal{NL}$ 中其他问题可在对数空间内归约到st-CONN)。□

$\phi_G(u,v,2l)$ 的递归计算, $l \geq 1$

对 $w=1,2,\dots,|V|$, (保存顶点名)

计算 $\sigma \leftarrow \phi_G(u,w,l)$, (由递归调用)

计算 $\sigma \leftarrow \sigma \wedge \phi_G(w,v,l)$, (由第二次递归调用)

如果 $\sigma=1$, 则返回1。 (成功:找到了一个中间结点)

结束 (扫描结束)

返回0 (如果扫描结束且未成功,则到这一步)

图 5.2 $\mathcal{NL} \subseteq D_{\text{SPACE}}(O(\log^2))$ 的递归程序

^① 类似地,计算 $\phi_G(u,v,2l+1)$ 时,可以扫描 G 中所有顶点 w ,并对每个 w ,检查是否 $\phi_G(u,w,l+1)=1$ 和 $\phi_G(w,v,l)=1$ 同时成立。

思考：定理 5.12 的证明主要依赖于两个观察。第一个观察是两个布尔条件的合取（或析取）可在空间 $s + O(1)$ 内验证，其中 s 是验证一个条件的空间复杂度。利用简单组合（即引理 5.1）可以证明这一点。第二个观察是第一个观察的推广：它宣称一个存在断言（或一个通用的量化断言）可以通过扫描相关域中所有可能的值来验证（并对每个值检查该断言），这会造成一个额外的空间复杂度负载，它是域规模的对数。

当考虑一个具体而简单的计算问题如 st-CONN 时，证明定理 5.12 是很方便的。不过，同样的思想也可以直接应用于 $\mathcal{NL}$ （或任意 N_{SPACE} 类）。

将 $\mathcal{NL}$ 类置于 $\mathcal{NC}^2$ 中

从 st-CONN 的简单定义，可以很方便地将 $\mathcal{NL}$ 置于复杂性类如 $\mathcal{NC}^2$ 中（即可由一致的、对数平方深度和有限扇入的电路类来判定）。只需观察到 st-CONN 可以通过将一个非奇异矩阵（即邻接矩阵中对角线上元素均为 1）提升到足够高的幂次（即其维数）。矩阵的求平方运算可以用对数深度有限扇入的一致电路类（即 $\mathcal{NC}^1$ 中电路）来计算，而重复对 n 行 n 列矩阵的 n 次幂求平方，可以利用一致性的、多项式规模和深度为 $O(\log^2 n)$ 的有限扇入电路类来计算；因此， $\text{st-CONN} \in \mathcal{NC}^2$ 。事实上，注意到 st-CONN 实际上表示任意的 $\mathcal{NL}$ 计算，从而有 $\mathcal{NL} \subseteq \mathcal{NC}^2$ （或注意到任意对数空间归约可用对数深度有限扇入的一致电路类来计算）。

5.3.2.3 补问题或 $\mathcal{NL} = \text{coNL}$

回顾（合理的）非确定时间复杂性类在补运算下并不是封闭的。进一步地，人们普遍认为 $\mathcal{NP} \neq \text{coNP}$ 。相反，（合理的）非确定的空间复杂性类对于补运算则是封闭的，这表现为结论 $\mathcal{NL} = \text{coNL}$ ，其中 $\text{coNL} \stackrel{\text{def}}{=} \{ \{0,1\}^* \setminus S : S \in \mathcal{NL} \}$ 。

在证明 $\mathcal{NL} = \text{coNL}$ 之前，注意到，这个结论的证明等价于提出一种 st-CONN 到其补问题的对数空间 Karp-归约（或等价地，一个反方向的归约，见习题 5.21）。我们的证明从另一种角度看 $\mathcal{NL}$ -vs- coNL 问题，根据 $\mathcal{NL}$ 与 $\mathcal{NL} \cap \text{coNL}$ 之间的关系（见习题 2.37）来重述该问题，并为 $\mathcal{NL} \cap \text{coNL}$ 类提供一个“操作性的解释”。

回顾 $\mathcal{NL}$ 的定义，如果存在一个非确定的对数空间机器 M 可以接受集合 S ，且被非确定机器接受的条件本质上是非对称的，则称集合 S 属于 $\mathcal{NL}$ 。就是说， $x \in S$ 蕴含着存在 M 对于输入 x 的可接受计算，而 $x \notin S$ 蕴含着 M 对 x 的所有计算都是不可接受的。因此， M 对于输入 x 的可接受计算的存在性明确表明 $x \in S$ ，而 M 对于输入 x 的不可接受计算的存在性并不一定表示 $x \notin S$ 。相反，当 $S \in \mathcal{NL} \cap \text{coNL}$ 时，则对 $x \in S$ 和 $x \notin S$ （或等价地，对 $x \in \bar{S} \stackrel{\text{def}}{=} \{0,1\}^* \setminus S$ ）都存在一个明确的指示，而这两种类型的指示可由不同的非确定机器提供（即或者接受 S ，或者接受 $\bar{S}$ ）。将两个机器结合起来，可以得到一个非确定机器，它对所有输入，有时会输出正确答案，并且要么输出正确答案，要么输出一个特殊符号（“不知道”）。这样就得到如下定义，其中将布尔函数看作一种特例。

定义 5.13（函数的非确定计算） 称一个非确定的机器 M 计算函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ ，如果对任意 $x \in \{0,1\}^*$ ，以下两个条件成立。

- (1) M 对于输入 x 的每个计算都得到一个 $\{f(x), \perp\}$ 中的输出，其中 $\perp \notin \{0,1\}^*$ 是一个特殊符号（表示“不知道”）；
- (2) 存在 M 对于输入 x 的一种计算，可以得到输出 $f(x)$ 。

注意到, $S \in \mathcal{NL} \cap \text{co}\mathcal{NL}$ 当且仅当存在一个非确定的对数空间机器, 可以计算 S 的特征函数 (见习题 5.25)。回顾 S 的特征函数, 记作 χ_S , 是一个布尔函数, 满足 $x \in S$ 时, $\chi_S(x) = 1$, 否则, $\chi_S(x) = 0$ 。从而有, $\mathcal{NL} = \text{co}\mathcal{NL}$ 当且仅当对任意 $S \in \mathcal{NL}$, 存在一个计算 χ_S 的非确定对数空间机器。

定理 5.14 ($\mathcal{NL} = \text{co}\mathcal{NL}$) 对任意 $S \in \mathcal{NL}$, 存在一个计算 χ_S 的非确定对数空间机器。

与定理 5.12 情况相同, 这个结论可以推广到任意在限定空间内构造的至少为对数的函数 $s: \mathbb{N} \rightarrow \mathbb{N}$, 也就是说, 对这样的 s 以及每个 $S \in N_{\text{SPACE}(s)}$, 有 $\bar{S} \in N_{\text{SPACE}(O(s))}$ 。为证明这种推广, 可对以下证明进行扩展, 或使用一个充分的上行转换引理。

证明概要: 与定理 5.12 的证明相同, 只要构造一个非确定的对数空间机器, 能计算 st-CONN 的特征函数, 记作 χ , 即可 (即如果 G 中存在 s 到 t 的有向路径, 则 $\chi(G, s, t) = 1$, 否则, $\chi(G, s, t) = 0$)。

首先证明, χ 的计算可在对数空间内归约为由给定图上的给定顶点出发, 判定 (由一条有向路径) 可达顶点数量的问题。对于输入 (G, s, t) , 归约过程先求 s 在图 G 中的可达顶点数量, 然后从 G 中去掉 t , 得到图 G' , 求 s 在图 G' 中的可达顶点数量, 将两个数字相比较。显然当且仅当 (在图 G 中) 由 s 出发可以到达 t 时, 两个数字不同。另一种归约利用了习题 5.28 中的询问。将这两个归约中任意一个与一个非确定的求可达顶点数量的对数空间机器相结合, 可以得到一个非确定的计算 χ 的对数空间机器。为证明这一点, 可以利用这些归约的非适应性, 或者利用目标问题的解具有对数长度的事实, 见习题 5.29。因此, 我们主要讨论由给定图中的给定顶点出发, 求可达顶点数量的非确定对数空间机器。

给定一个 n -顶点图 $G = (V, E)$ 及顶点 v , 考虑由 v 出发, 利用长度至多为 i 的路径可达的顶点集合。将这个集合记作 R_i , 并观察到 $R_0 = \{v\}$, 且对任意 $i = 1, 2, \dots$, 有

$$R_i = R_{i-1} \cup \{u : \exists w \in R_{i-1}, \text{ s.t. } (w, u) \in E\} \quad (5.1)$$

我们的目标是在对数空间内 (非确定地) 计算 $|R_n|$ 。这可以利用 n 次迭代完成, 在第 i 次迭代中计算 $|R_i|$ 。在计算 $|R_i|$ 时需要借助于已知的 $|R_{i-1}|$, 这意味着需要存储 $|R_{i-1}|$ 。注意: 一旦完成了 $|R_i|$ 的计算, 就马上丢弃 $|R_{i-1}|$, 并保存 $|R_i|$ 。因此, 在第 i 次迭代中, 对以往迭代的记录只包含 $|R_{i-1}|$ 。

$|R_i|$ 的计算

给定 $|R_{i-1}|$, 非确定地计算 $|R_i|$ 时, 首先猜测 ($|R_i|$ 的值), 记作 g , 并用如下方法验证猜测的正确性。

(1) 直接验证是否 $|R_i| \geq g$ 。即以正常顺序扫描 V , 检查 g 个顶点中每一个是否属于 R_i 。也就是说, 在扫描中, 非确定地选择 g 个顶点, 对每个选中的顶点 w , 验证 w 是否由 v 经过长度不超过 i 的路径可达, 而这个检查过程在实现时, 只需要做出猜测并验证一条充分的路径即可 (见习题 5.19)。

我们利用 $\log_2 n$ 个比特来存储已被验证属于 R_i 的顶点数量, 另外 $\log_2 n$ 个比特保存当前扫描的顶点 (即 w), 还有 $O(\log n)$ 个比特用于验证是否存在从 v 到 w 的长度不超过 i 的路径。

(2) 对于条件 $|R_i| \leq g$ 的验证过程 (等价地, $|V \setminus R_i| \geq n - g$) 是程序中真正有价值的部分。事实上, R_i 的成员验证是容易的, 但这里我们希望验证非 R_i 的成员。这可以建立在已知 $|R_{i-1}|$ 的基础上, 从而需要考虑对 R_{i-1} 元素的非确定性枚举, 这又要求对非 R_i 成员的证明 (利

用式(5.1)可知。细节如下(图 5.3 给出了一种更加结构化的描述)。(再次)扫描 V , 对 $n-g$ 个(猜测的)顶点, 验证其不属于 R_i (即从 v 出发, 经长度不超过 i 的路径不可达)。由式(5.1)可知, 对 $u \notin R_i$ 的验证可归结为证明对每个 $w \in R_{i-1}$, 有 $u \neq w$ 且 $(w, u) \notin E$ 。如前面所提示的, 已知 $|R_{i-1}|$ 时, 可以枚举 R_{i-1} 中的元素, 从而只需对 R_{i-1} 中每个顶点检查上述条件是否满足即可。因此, 验证 $u \notin R_i$ 的过程如下。

① 扫描 V , 猜测 R_{i-1} 中的 $|R_{i-1}|$ 个顶点, 并对每个猜测直接验证条件是否满足(如第一步)。^①

② 对每个已猜测并在①中验证的 $w \in R_{i-1}$, 验证是否有 $u \neq w$ 以及 $(w, u) \notin E$ 同时成立。

由式(5.1)可知, 如果 u 通过了上述验证, 则必有 $u \notin R_i$ 。

我们利用 $\log_2 n$ 个比特来保存已被验证属于 $V \setminus R_i$ 的顶点数量, 另外 $\log_2 n$ 个比特保存当前顶点 u , 还有 $\log_2 n$ 个比特对已经验证属于 R_{i-1} 的顶点进行计数, 再用 $\log_2 n$ 个比特保存这样的顶点 w , 以及 $O(\log n)$ 个比特来验证 $w \in R_{i-1}$ (如第一步)。

如果上述任意一个验证过程失败, 则程序停止, 输出一个表示“不知道”的符号 $\perp$, 否则, 输出 g 。

```

Given  $|R_{i-1}|$  and a guess  $g$ , the claim  $g \geq |R_i|$  is verified as follows.

Set  $c \leftarrow 0$ .                                     (initializing the main counter)
For  $u = 1, \dots, n$  do begin                         (the main scan)
    Guess whether or not  $u \in R_i$ .
    For a negative guess (i.e.,  $u \notin R_i$ ), do begin
        (Verify that  $u \notin R_i$  via Eq. (5.1).)
        Set  $c' \leftarrow 0$ .                         (initializing a secondary counter)
        For  $w = 1, \dots, n$  do begin                 (the secondary scan)
            Guess whether or not  $w \in R_{i-1}$ .
            For a positive guess (i.e.,  $w \in R_{i-1}$ ), do begin
                Verify that  $w \in R_{i-1}$  (as in Step 1).
                Verify that  $u \neq w$  and  $(w, u) \notin E$ .
                If some verification failed
                    then halt with output  $\perp$  otherwise increment  $c'$ .
            End (of handling a positive guess for  $w \in R_{i-1}$ ).
        End (of secondary scan).                     ( $c'$  vertices in  $R_{i-1}$  were checked)
        If  $c' < |R_{i-1}|$  then halt with output  $\perp$ .
        Otherwise ( $c' = |R_{i-1}|$ ), increment  $c$ .    ( $u$  verified to be outside of  $R_i$ )
    End (of handling a negative guess for  $u \notin R_i$ ).
End (of main scan).                                 ( $c$  vertices were shown outside of  $R_i$ )
If  $c < n - g$  then halt with output  $\perp$ .
Otherwise  $n - |R_i| \geq c \geq n - g$  is verified.
    
```

图 5.3 证明 $NL = coNL$ 的主要步骤

显然, 上述的非确定程序使用了对数空间。可以证明, 当给定了 $|R_{i-1}|$ 的正确值后, 该程序能非确定地计算 $|R_i|$ 的值。也就是说, 如果所有的验证都通过了, 则一定有 $g = |R_i|$, 而如果 $g = |R_i|$, 则存在足够多的非确定选择, 能满足所有的验证。

回顾 R_n 是迭代计算的, 从 $|R_0| = 1$ 开始, 根据 $|R_{i-1}|$ 来计算 $|R_i|$ 。对于 $i = 1, 2, \dots, n$, 每一步

① 注意: ①中蕴含了一个非确定程序, 可以计算映射 $(G, v, i, |R_{i-1}|) \rightarrow R_i$, 其中 R_{i-1} 表示 G 中从 v 出发, 经由长度不超过 i 的路径可达的顶点集。

迭代都是非确定的，并且在结束时或者返回正确的 $|R_i|$ （此时， $|R_{i-1}|$ 被丢弃），或者验证失败并停机（此时，整个程序中止并输出 $\perp$ ）。这样就得到了一个计算 $|R_n|$ 的非确定对数空间机器，定理得证。□

思考：第2步是（定理 5.14）整个证明的核心。这一步使用一个非确定的程序来验证 NL-型集合的非成员。事实上，NL-型集合的非成员验证是非确定程序能够解决的典型任务（即它们被定义为适合解决这些任务），从而第1步是相当直截了当的。相反，非成员的非确定验证并不是一种普遍现象，因此，第2步就不那么直接。尽管如此，在本节（第2步中）非成员的验证也是由迭代程序实现的，该程序使用的资源控制在允许范围内（即对数空间）。

5.3.3 回顾与讨论

本节可以看作是对“非确定计算的功能”（这是一个有点自相矛盾的名词）的研究。回顾非确定性的程序实际上被看作是假想中的抽象，目的是理解一些基本现象如对证明的验证（见 2.1.5 节）。由于这些假想的抽象是时间复杂度最基本的要求，人们可能希望通过比较来更深入的理解，具体而言，就是在空间复杂度中研究非确定性。进一步地，我们会发现，非确定的空间受限机器可以引出一些有趣的计算现象。

本节似乎实现了上述期望。例如， $NL = coNL$ 的事实，而通常的猜测是 $NP \neq coNP$ ，因此本节表明这个猜测并不总是成立。不是说一个存在量词不能由一个通用量词“可行地替换”，而是这种替换的可行性在很大程度上依赖于可行性的定义。转向另一个目标，我们知道，st-CONN 可以在对数空间内 Karp-归约为 st-unCONN（即在两个指定顶点间不存在直接路径的图的集合，见习题 5.21）。

然而， NL 类究竟代表什么（除了 st-CONN，这个问题可能不仅仅是该类中的完全问题，见 5.3.2.1 节）。回到 5.3.1 节，其中表明 $N_{SPACEoff-line}$ 类直接体现了空间受限验证的概念。在该模型（称为离线模型）中，要验证的证明写在一个特殊设备上（类似于被其证明的断言），并可自由地读出。相反，另一个类 $N_{SPACEon-line}$ 中体现了如何通过串行读取（而非反复扫描）来验证证明。在这种情况下，如果验证程序（在未来某个时刻）需要重新检查当前读取的部分证明，则它必须保存这些部分（并付出相应的存储代价）。因此， $N_{SPACEon-line}$ 中体现的在线模型需要以串行方式读取证明过程，并为了将来验证而做记录，而不是反复扫描整个证明。在线模型反映了做记录的真正空间复杂度，因此，可以串行地验证证明。事实上（如 5.3.1 节所述），我们认为，离线模型所需要的临时存储空间是不合理的，所需要的证明无形中也是很长的。

5.4 PSPACE 及游戏

如 5.2 节所述，人们遇到的绝大多数计算问题都无法在少于对数空间内求解，另一方面，人们也很少遇到所需空间超过了多项式空间的计算问题。可在多项式空间内求解的判定问题记作 $PSPACE \stackrel{\text{def}}{=} \bigcup_c D_{SPACE}(p_c)$ ，其中 $p_c(n) = n^c$ 。

为了更深入地理解 $PSPACE$ 的功能，注意到 $PH \subseteq PSPACE$ ，例如，一个多项式空间算法可以很容易地验证定义 3.8 中的量化条件。事实上，这样的算法可以有无穷多次量词转换（见以下的定理 5.15）。另一方面，由定理 5.3， $PSPACE \subseteq EXP$ ，其中

$EXP = \bigcup_c D_{TIME}(2^{p_c}), p_c(n) = n^c$ 。

$PSPACE$ 类可以解释为在某种高效的两方游戏中确定胜利者的复杂度，具体来讲，就是 3.2.1 节中的游戏。这里所说的两方游戏，要满足以下 3 个条件。

(1) 参与方轮流行动，影响游戏的（全局）位置，其中描述每个行为的串长度的上限是初始位置长度的多项式。

(2) 基于前一个位置以及当前参与方的行为来更新当前位置。更新所需的时间是初始位置长度的多项式（等价地，可能需要一个多项式时间更新程序，并假设当前位置长度的上限是初始位置长度的多项式）。

(3) 每个位置中的胜利者可在多项式时间内确定。

回顾对任意给定的 k ，我们已经（在 3.2.1 节）证明了 Σ_k 对应于上述游戏中确定 k -次行为取胜策略（对第一个参与方）的存在性这一问题。同样的对应关系还存在于 $PSPACE$ 与（在上述类型游戏中）判定是否存在多项式次行为后的取胜策略的问题。也就是说，一方面，将使第一方具有 $\text{poly}(|x|)$ 次行为取胜策略的初始位置集合记作 x ，则 x 属于 $PSPACE$ ；另一方面，由上述的定理 5.15， $PSPACE$ 中每个集合可以看作是（在一个适当游戏中）使第一方具有多项式次行为取胜策略的初始位置集合。实际上，这里的对应关系存在于取胜策略的判定与量化布尔公式（QBF）的可满足性判定之间，见习题 5.30。

QBF 与 PSPACE

量化布尔公式为公式（如 SAT 中的 CNF 范式）中出现的每个变量附加了量词（注意：与习题 3.7 不同，这里对存在量词与全称量词之间的转换次数没有任何限制。进一步的讨论可参考附录 G.2）。前面有注释，判定量化布尔公式的可满足性（QBF）属于 $PSPACE$ 类。下面证明 $PSPACE$ 中每个问题都可 Karp-归约到 QBF。

定理 5.15 在多项式时间多到一映射下，QBF 是 $PSPACE$ 完全的。

证明：如前所述，QBF 可由对量化公式求值的多项式空间算法求解。具体地，考虑一个递归程序，在对两个剩余公式求值之后，可消除一个量词，并注意到第一次（递归）求值中使用的空间可以在第二次计算中重复利用（另外，考虑如 5.1.4 节中 DFS-类型的程序）。注意：使用的空间是递归深度的线性函数，而递归深度又是输入公式长度的线性函数。

现在我们证明任意集合 $S \in PSPACE$ 可以多到一地归约为 QBF。证明过程类似于定理 5.12（其中证明了 $NL \in D_{SPACE}(\log^2)$ ），但在这里使用了一个不太明确的图（见习题 5.21，而非一个明确给定的图）。具体地，我们考虑由（判定 S 中成员的算法 A 的）即时配置构成的有向图，这里配置的概念有所不同，其中包含了整个输入。就是说，在证明的其余部分，配置中包含了算法使用的所有存储设备（含输入设备）的内容，以及算法在每个设备上的位置。因此，对于（归约的）输入 x ，要考虑有向图 $G = G_{x,A} = (V_x, E_A)$ ，其中 V_x 表示输入 x 所有可能的配置，而 E_A 表示算法 A 的迁移函数（即 A 中单个步骤的效果）。

与定理 5.12 的证明相同，对于图 G ，如果在 G 中存在从 u 到 v 的长度不超过 l 的路径，则定义 $\phi_G(u, v, l) = 1$ （否则， $\phi_G(u, v, l) = 0$ ）。设 s 表示对 $A(x)$ 初始配置的编码， t 表示对可被规范接受的配置的编码，我们需要确定 $\phi_G(s, t, 2^m)$ ，其中 G 取决于算法 A ， $m = \text{poly}(|x|)$ ，使得 $A(x)$ 使用的空间至多为 m 且运行步骤不超过 2^m 步。由配置的具体定义（其中包括所有相关信息，输入 x 也在内），根据固定算法 A 易确定 $\phi_G(u, v, l)$ 的值（即或者 $u = v$ ，或者 v 是 u

的后续配置)。回顾 $\phi_G(u, v, 2l) = 1$ 表示当且仅当存在一个配置 w , 使得 $\phi_G(u, w, l) = 1$ 和 $\phi_G(w, v, l) = 1$ 同时成立。因此, 得到递归式

$$\phi_G(u, v, 2l) = \exists w \in \{0, 1\}^m \phi_G(u, w, l) \wedge \phi_G(w, v, l) \quad (5.2)$$

其中递归的基础 (即 $\phi_G(u, v, 1)$) 是一个简单的命题公式 (见前面的说明)。式 (5.2) 中存在的问题是 $\phi_G(\cdot, \cdot, 2l)$ 的表达式中 $\phi_G(\cdot, \cdot, l)$ 出现了两次, 这使得递归构造的公式长度加倍 (从而呈指数增长)。

我们的目标是在表示 $\phi_G(\cdot, \cdot, 2l)$ 时只使用一次 $\phi_G(\cdot, \cdot, l)$ 。这个附加限制能防止指数增长, 它相当于在计算 $\phi_G(u, v, 2l)$ 时, 要重复利用计算两次 $\phi_G(\cdot, \cdot, l)$ 所使用的空间。主要思路是将条件 $\phi_G(u, w, l) \wedge \phi_G(w, v, l)$ 替换为条件 “ $\forall (u', v') \in \{(u, w), (w, v)\} \phi_G(u', v', l)$ ” (其中量化过程在一个不同于 $\{0, 1\}$ 的二元素集合上进行)。下一步, 要利用附加的量词和某些简单布尔条件来重新定义非标准的量词 (作用于一对特殊的串)。即将非标准量词 $\forall (u', v') \in \{(u, w), (w, v)\}$ 用标准量词 $\forall \sigma \in \{0, 1\} \exists u', v' \in \{0, 1\}^m$ 和以下的辅助条件来替代:

$$[(\sigma = 0) \Rightarrow (u' = u \wedge v' = w)] \wedge [(\sigma = 1) \Rightarrow (u' = w \wedge v' = v)] \quad (5.3)$$

从而, $\phi_G(u, v, 2l)$ 成立当且仅当存在 w , 使得对任意 σ , 存在 (u', v') 同时满足式 (5.3) 和 $\phi_G(u', v', l)$ 。注意: $\phi_G(\cdot, \cdot, 2l)$ 的这种表达式的长度等于 $\phi_G(\cdot, \cdot, l)$ 的长度加上一个附加项 $O(m)$ 。因此, 利用递归构造, 公式的长度随递归次数的增加只呈线性增长。

递归本身将 (S) 实例 x 映射为量化布尔公式 $\Phi(s_x, t, 2^m)$, 其中 s_x 表示 $A(x)$ 的初始配置 (t 和 $m = \text{poly}(|x|)$ 的定义同上), 用如下方式递归地定义 Φ 。

$$\begin{aligned} \Phi(u, v, 2l) \stackrel{\text{def}}{=} & \exists w \in \{0, 1\}^m \forall \sigma \in \{0, 1\} \exists u', v' \in \{0, 1\}^m \\ & [(\sigma = 0) \Rightarrow (u' = u \wedge v' = w)] \\ & \wedge [(\sigma = 1) \Rightarrow (u' = w \wedge v' = v)] \\ & \wedge \Phi(u', v', l) \end{aligned} \quad (5.4)$$

$\Phi(u, v, 1) = 1$ 当且仅当或者 $u = v$ 或者存在 u 到 v 的边。注意: $\Phi(u, v, 1)$ 是一个 (固定的) 命题公式, 其中的布尔变量代表 u 和 v 中的比特, 使得 $\Phi(u, v, 1)$ 是可满足的, 当且仅当或者 $u = v$, 或者在 A 的计算中, v 是 u 的后续配置。另一方面, 注意到 $\Phi(s_x, t, 2^m)$ 是一个量化公式, 其中的 s_x 、 t 和 m 是固定的, 且并未出现量化变量。

我们强调, 从 x 到 $\Phi(s_x, t, 2^m)$ 的映射可在多项式时间内计算。首先, 注意命题公式 $\Phi(u, v, 1)$, 其中的布尔变量是 u 和 v 中的比特, 它们表示非常简单的条件, 并且一定可以在多项式时间内构造 (即布尔变量个数的多项式, 等于 $2m$)。其次要注意的是, 给定 $\Phi(u, v, l)$, 其中 $(l > 1)$ 包含不出现的量化变量, 构造 $\Phi(u, v, 2l)$, 只需要替换变量名称、增加量词, 并增加如式 (5.4) 所定义的递归中的布尔条件即可。这肯定可以在多项式时间内完成。最后, 注意 $\Phi(s_x, t, 2^m)$ 的构造主要依赖于 x 的长度, 而 x 自身又只影响 s_x (并以平凡方式影响)。回顾 $m = \text{poly}(|x|)$, 从而上述所有过程都可在 $|x|$ 的多项式时间内完成。因此, 给定 x , $\Phi(s_x, t, 2^m)$ 可在多项式时间内构造。

最后, 注意到 $x \in S$ 当且仅当 $\Phi(s_x, t, 2^m)$ 是可满足的。定理得证。 ■

其他 $PSPACE$ -完全问题

本节一开始提到, $PSPACE$ 与各种游戏中取胜策略的确定紧密相关。建立这种关系需要考虑对应于 QBF 的可满足性的一般游戏(见习题 5.30)。 $PSPACE$ 与确定游戏取胜策略之间的关系比这个一般游戏中所表现的还要密切: 在一些自然的(推广的)游戏中确定取胜策略也是 $PSPACE$ -完全的(见参考文献[208]中 8.3 节)。这使本节的标题更加合理化。

本章注释

本章内容建立于将“经典”结论(20 世纪 70 年代或更早被证明的)与“现代”结论(20 世纪 80 年代晚期被证明的)相混合的基础上。我们希望强调某些问题与其解的定义之间存在着时间缝隙。细节如下。

首先介绍“经典”结论。其中包括 st-CONN 的 NL -完全性, 确定的空间受限机器可以模拟非确定的空间受限机(即 Savitch^[197]证明的定理 5.12), QBF 的 $PSPACE$ -完全性, 以及电路深度与空间复杂度之间的关系(见 5.1.4 节及习题 5.12, 来自于 Borodin^[48])。

在介绍“现代”结论之前, 要指出一些研究人员可能会对这样一种印象感到气馁, 即“数十年来, 研究者都无法解决复杂性理论中任何一个著名的公开问题”。我们认为, 这种印象根本就是错误的。具体地, 数十年来, 除了对许多基本问题的理解有了根本性进展之外, 这些研究者似乎还忘记了, 某些著名的公开问题实际上已经得到解决。本章就给出了这样两个例子。

20 年来, $NL=coNL$ 是否成立一直是一个著名的公开问题。另外, 该问题与形式语言的早期(即 20 世纪 50 年代)研究中某些更古老的公开问题有关。^①这一公开问题在 1988 年被 Immerman^[121]和 Szelepcsényi^[219]解决, 他们(分别独立地)证明了定理 5.14(即 $NL=coNL$)。

在超过 20 年的时间里, 无向连通性(UCONN)是关于随机性计算能力的最著名例子之一。回顾经典的线性时间(确定)算法(如 BFS 和 DFS)要求大量使用临时存储空间(即是图规模的线性函数)。另一方面, (1979 年以来, 见 6.1.5.2 节)已知多项式长度的随机漫游会以极高概率经过所有顶点(在相应的连通分量中)。因此, 得到的解 UCONN 的随机算法使用了最少的临时存储空间(即是图规模的对数)。在 20 世纪 90 年代早期, 该算法(以及整个 BPL 类(见定义 6.11))可以在多项式时间和对数空间内去随机化(见定理 8.23), 但是, 虽然人们已经进行了十几年的尝试, 对于这种自然的、普遍存在的问题而言, 随机性的空间复杂度与确定的多项式时间算法之间仍存在明显的缝隙。这个缝隙被 Reingold[190]修补, 他在 2004 年证明了定理 5.6。^②我们的陈述(5.2.4 节)采用了 Reingold 的思想, 但是 5.2.4.2 节中的具体定义在参考文献[190]中并未出现。

习 题

5.1 (对输入带上超出输入部分的扫描) 设 A 为任意一个空间复杂度为 s 的算法。证明

① 特别地, 可由线性空间非确定性机器识别的集合类等价于对上下文敏感的语言类(见参考文献[123]中 9.3 节), 因此, 定理 5.14 解决了后者在补运算下的封闭性问题。

② 我们指出, 在同一时期, Trifonov^[224]使用完全不同方法, 独立地开发了一个近似对数空间的算法。

存在一个功能等价的算法 A' ，具有空间复杂度 $s'(n) = O(s(n) + \log n)$ ，且不会扫描输入带上超出输入的部分。

提示：证明对于输入 x ，算法 A 不扫描输入带上与输入距离超出 $2^{O(s(|x|))}$ 的部分。

(附加提示：考虑 $A(x)$ 的计算中，当 A 读取不属于输入的输入带上位置时的即时配置。)

5.2 (在只写输出带上的重写) 令 A 为任意一个空间复杂度为 s 的算法。证明存在一个功能等价的算法 A' ，从不在输出设备（同一位置）上重复写入，且 A' 的空间复杂度 s' 满足 $s'(n) = s(n) + O(\log l(n))$ ，其中 $l(n) = \max_{x \in \{0,1\}^n} |A(x)|$ 。

提示：算法 A' 迭代运行，在第 i 次迭代中，模拟 A 对于输入 x 的计算并输出 $A(x)$ 的第 i 个比特。对 A 的第 i 次模拟不用输出 $A(x)$ ，而是保持 $A(x)$ 的输出带上第 i 个位置的记录（并以输出这个比特的最终值而结束程序）。事实上，模拟过程需要保持 i 的当前值，以及被模拟机器（即 A ）在其输出带上的当前位置。

5.3 (双对数空间的功能) 对任意 $k \in \mathbb{N}$ ，令 w_k 表示所有 k -比特长串的级联（以词典序），中间被“*”分隔（即 $w_k = 0^{k-2}00*0^{k-2}01*0^{k-2}10*0^{k-2}11*\dots*1^k$ ）。证明集合 $S = \{w_k : k \in \mathbb{N}\} \subset \{0,1,*\}$ 不是规范的，从而可在双对数空间内判定。

提示： S 的不规范性可利用标准技术证明。目的是要开发一个算法，注意到 $|w_k| > 2^k$ ，从而 $O(\log k) = O(\log \log |w_k|)$ 。为判定 x 是否为 S 的成员，可以重复检查是否有 $x = w_i$ ， $i = 1, 2, \dots$ ，当检查到一个明显结果（即或者验证了 $x = w_i$ ，或者对每个 $k \geq i$ ，检查到 $x \neq w_k$ 的证据）时，算法停止。利用 $(w_i \text{ 中的 })^*$ ，可在空间 $O(\log i)$ 内完成第 i 次迭代。进一步地，对于输入 $x \notin S$ ，算法在至多 $\log |x|$ 次迭代后停止并丢弃这个输入。实际上，处理另一个相关的集合 $\{w_1 ** w_2 ** \dots ** w_k : k \in \mathbb{N}\}$ 会更容易；另外，此时，可以省略 w_i 中（以及这些 w_i 之间）所有的“*”。

5.4 (小于双对数空间的弱点) 证明，对于 $l(n) = \log \log n$ ，有 $D_{\text{SPACE}}(o(l)) = D_{\text{SPACE}}(O(1))$ 。

提示：令 s 表示机器的（二元）空间复杂度。证明，如果 s 是无界的，则必有 $s(n) = \Omega(\log \log n)$ 。具体地，对每个整数 m ，考虑一个最短串 x ，满足对于输入的 x ，机器使用的空间至少是 m 。对于输入带上的每个位置，考虑机器的剩余配置序列（即其临时存储单元中的内容），^①使得序列中第 i 个元素表示机器在时刻 i 经过这个输入位置的剩余配置。

对于初学者，注意到这个“经过序列”的长度上限是可能的剩余配置数量，它不超过 $t = 2^{s(|x|)} \cdot s(|x|)$ 。因此，这种经过序列的数量上限为 t' 。现在，如果 $t' < |x|/2$ ，则存在 3 个输入位置，具有相同的经过序列，且其中的两个取值相同。提取这两个位置中的串，可以得到一个更短的输入，使机器在其上的行为完全相同，这就与 x 是机器使用至少为 m 的空间所能处理的最短输入相矛盾。从而，必有 $t' \geq |x|/2$ ，以及对于无穷多个 x ，有 $s(|x|) = \Omega(\log \log |x|)$ 。

5.5 (空间复杂度与停机) 续定理 5.3，证明对任意具有（二元）空间复杂度 s 的算法 A ，存在一个空间复杂度为 $s'(n) = O(s(n) + \log n)$ 的算法 A' ，对每个输入都停机，且对每个使 A 停机的 x ，有 $A'(x) = A(x)$ 。

^① 注意：定理 5.3 的证明不同，机器在输入带上的位置并不是配置的一部分。另一方面，虽然没有明确指出，但配置中也包括了机器在存储带上的位置。

提示: 对于输入 x , 算法 A' 模拟 $A(x)$ 的最多 $t(|x|)+1$ 步, 这里 $t(n) = n \cdot 2^{s(n)+\log_2 s(n)}$ 。

5.6 (一些对数空间算法) 针对以下计算问题, 提出对数空间算法。

(1) 将一对给定整数相加及相乘。

提示: 利用引理 5.2, 首先将输入转化为一种更方便的形式, 然后执行操作, 最后将结果转化为适当形式。例如, 在将 $x = \sum_{i=0}^{n-1} x_i 2^i$ 和 $y = \sum_{i=0}^{n-1} y_i 2^i$ 相加时, 一种方便的形式是 $((x_0, y_0), \dots, (x_{n-1}, y_{n-1}))$ 。

(2) 判定两个给定的串是否相同。

(3) 在给定串 $s \in \{0,1\}^*$ 中, 找出所有给定样式 $p \in \{0,1\}^*$ 。

(4) 将图的邻接矩阵表示转化为依附列表表示, 反之亦然。

(5) 判定是否给定的图为无圈图 (即没有简单回路)。

提示: 考虑如下对图的扫描过程。输入第 i 条边依附的一个顶点 v , 如果 v 的度数至少是 $i+1$, 则由其第 $i+1$ 条边离开该顶点, 否则, 由第 1 条边离开。注意: 如果这里是从任何一棵树的任意顶点开始, 则扫描过程构成一个 DFS。另一方面, 对每个有圈图, 存在一个顶点 v 及依附于 v 的边 e , 使得上述扫描过程从 v 开始, 通过边 e , 最后从另一条不同于 e 的边回到 v 。

(6) 判定给定的图是否为树。

提示: 图 $G=(V,E)$ 是一棵树当且仅当它是无圈的且 $|E|=|V|-1$ 。

5.7 (另一个合成结论) 续 5.1.3.3 节中的讨论, 证明当给定关于 Π' 的 (l, l') -限制预言访问, 且 Π' 可在空间 s_2 内求解时, 如果 Π 可在空间 s_1 内求解, 则 Π 也可在空间 s 内求解, 使得 $s(n) = 2s_1(n) + s_2(l(n)) + 2l'(n) + \delta(n)$, 其中 $\delta(n) = O(\log(l(n) + l'(n) + s_1(n) + s_2(l(n))))$ 。特别地, 如果 s_1 、 s_2 和 l' 至多为对数, 则 $s(n) = O(\log n)$, 因为 (由习题 5.10), 此时 l 至多为多项式。

提示: 可将有预言机帮助的 Π 的运算看作是由迭代构成, 在第 i 次迭代中, 在得到第 $i-1$ 个应答 (记作 w_{i-1}) 之后, 基于初始输入 (记作 x)、第 $i-1$ 个预言应答 (记作 a_{i-1}) 以及工作带在 $i-1$ 时刻的内容来确定第 i 个询问 (记作 q_i)。注意: 映射 $(x, a_{i-1}, w_{i-1}) \rightarrow (q_i, w_i)$ 可以利用临时存储中的 $s_1(|x|) + \delta(|x|)$ 个比特来计算, 因为预言机影响着该映射 (当 x , a_{i-1} 和 w_{i-1} 位于不同设备上时)。将每次迭代与 Π' 的运算相结合 (利用引理 5.2 的一种变形), 我们得到结论, 映射 $(x, a_{i-1}, w_{i-1}) \rightarrow (a_i, w_i)$ 可在空间 $s_1(n) + s_2(l(n)) + O(\log(l(n) + s_1(n) + s_2(l(n))))$ 内计算 (不用保存临时的中间数据 q_i)。因此, 可以利用空间 $s(n)$ 来模拟整个计算, 而额外的 $s_1(n) + 2l'(n)$ 比特空间用于保存预言机工作带的内容, 以及第 $i-1$ 个和第 i 个预言应答。

5.8 (非适应的归约) 续 5.1.3.3 节中的讨论, 定义非适应的空间受限归约如下。首先, 对任意问题 Π' , 定义 (“直积”) 问题 $\overline{\Pi'}$, 使得 $\overline{\Pi'}$ 的实例为 Π' 实例构成的序列。序列 $\overline{y} = (y_1, y_2, \dots, y_t)$ 是 (关于问题 $\overline{\Pi'}$ 的) 实例 $\overline{x} = (x_1, x_2, \dots, x_t)$ 的有效解当且仅当对任意 $i \in [t]$, y_i 是 (问题 Π' 的实例) x_i 的有效解。现在, 从 Π 到 Π' 的非适应性归约定义为由 Π 到 $\overline{\Pi'}$ 的单一询问归约。

(1) 注意：该定义允许预言机自由地扫描应答序列（即可以在保存着不同应答的块之间自由移动）。尽管如此，仍要证明这样不会使机器的功能有所增加（与一个以“半单向”方式读取预言应答设备的机器相比（即在读后续应答之后，决不会返回来读某个先前应答中的比特））。也就是说，证明一个空间复杂度为 s 的普通的非适应归约可以用空间复杂度为 $O(s)$ 的非适应归约来模拟，在后一种归约中，得到预言应答 $(y_1, y_2, \dots, y_l)$ 之后，只有在读 $y_{i+1}, y_{i+2}, \dots, y_l$ 中的任意比特之前，才可以读 y_i 中的比特。

提示：将询问序列 $\bar{x} = (x_1, x_2, \dots, x_l)$ 用询问序列 $(\bar{x}, \bar{x}, \dots, \bar{x})$ 代替，后者重复的次数为 $2^{O(s)}$ 。

(2) 证明如果 Π 可以非适应地在空间复杂度 s_1 内归约到 Π' ，归约中至多发出 l 个询问且 Π' 的空间复杂度为 s_2 ，则 Π 可在空间 s 内求解，满足 $s(n) = O(s_1(n) + s_2(l(n)))$ 。作为热身，先考虑一般的单询问归约（从 Π 到 Π' ）情形。

提示：合成之后，对于输入 x 的运算，可以写成 $E(x, \bar{A}(G(x)))$ ，其中 G 代表询问生成阶段， $\bar{A}$ 代表将解 Π' 的算法应用于询问序列中的每个串，而 E 代表求值阶段。利用引理 5.2（的变形）分析该计算的空间复杂度。

5.9 针对 5.1.3.3 节中的讨论，证明对任意 s ，任意空间复杂度为 s 的问题可以利用常数空间 $(2s, 2s)$ -限制的归约，归约到一个常数空间可解的问题。

提示：归约的目标是与该算法（空间复杂度为 s ）相关的“下一配置函数”，这里的配置也包含机器当前检查的输入中的单个比特（即机器在输入设备上的位置处存储的比特）。为了便于函数计算，可以选择一种适当的配置形式。注意预言机的大量运算是迭代地将预言应答带的内容（稍做修改并）复制到预言询问带上。

5.10 续 5.1.3.3 节，称一个归约是 $(\cdot, l')$ -限制的，如果存在某个函数 l ，使得归约是 (l, l') -限制的，就是说，在这个定义中，只对预言应答的长度进行限制。证明，任意空间复杂度为 s 的 $(\cdot, l')$ -限制归约也是 (l, l') -限制的，其中 $l(n) = 2^{O(s(n) + l'(n) + \log n)}$ 。实际上是要证明该归约的时间复杂度为 l 。

提示：考虑充分意义上的即时配置；具体地，该配置中既包含工作带和预言应答带的内容，也包含机器在这些带（以及输入带）上的位置。

5.11 （对数空间归约的传递性）证明对数空间 Karp-归约具有传递性。给出对数空间 Levin-归约的定义，并证明也具有传递性。

提示：利用引理 5.2，注意到这种归约仅仅是一些可利用对数空间计算的函数。

5.12 （对数空间一致的 $\mathcal{NC}^1$ 包含于 $\mathcal{L}$ 中）设问题 Π 可由一类有界扇入、深度为 d 的对数空间电路簇求解，满足 $d(n) \geq \log n$ 。证明 Π 可由任意一个空间复杂度为 $O(d)$ 的算法求解。

提示：将 5.1.4 节略述的算法与对数空间一致性的定义相结合（利用引理 5.2）。

5.13 （常数度、对数直径图上的 UCONN）构造一个对数空间算法，判定以下的承诺问题，其参数为常数 c 和 d 。称输入的图满足承诺，如果每个顶点的度数至多为 d ，且位于同一连通分量中的每对顶点由长度不超过 $c \log_2 n$ 的路径相连，其中 n 表示输入图的顶点数。算法的任务是判定输入的图是否为连通图。

提示：对图中的每对顶点，检查它们是否连通（或者，我们只需检查是否每个顶点与第一个顶点相连即可）。根据承诺，只要检查所有长度不超过 $l \stackrel{\text{def}}{=} c \log_2 n$ 的路径即可，而这些路

径可以用 $l \cdot \lceil \log_2 d \rceil$ 比特的存储空间来枚举。

5.14 (5.2.4.2 节的热身) 续 5.2.4.1 节, 构造 G_i 到 G_{i-1} 的对数空间变换。

提示: 给定图 G_i 作为输入, 可以用如下方法构造 G_{i-1} , 先构造 $G' = G_i^c$, 再构造 $G' \text{ ZZ } G$ 。为了构造 G' , 要扫描 G_i 中所有顶点 (将当前顶点保存于临时存储设备中), 并对每个顶点, 构造其在 G' 中的 “相距为 c 的邻居” (通过利用空间 $O(c)$ 来枚举所有可能的 “相距为 c 的邻居”)。类似地, 可以构造 $G' \text{ ZZ } G$ 中的顶点邻居 (通过保存当前顶点名, 并利用常数空间来指示在 G 中与其相关联的边)。

5.15 (st-UCONN) 续 5.2.4 节, 证明以下计算问题属于 $\mathcal{L}$ 类: 给定一个无向图 $G = (V, E)$ 及两个指定顶点, 确定 G 中是否存在 s 到 t 的路径。

提示: 注意: 在 5.2.4 节描述的转化易被扩展为 G_0 中顶点到 $G_{O(\log|V|)}$ 中顶点的映射, 而同时保持连通关系 (即 u 和 v 在 G_0 中是连通的当且仅当其在映射下的像在 $G_{O(\log|V|)}$ 中是连通的)。

5.16 (图的二部性) 证明在对数空间归约意义下, 判定输入的图是否为二部图 (即 2-可着色) 在计算上等价于 st-UCONN (定义见习题 5.15)。

提示: 两个归约都使用了图 $G = (V, E)$ 到二部图 $G' = (V', E')$ 的映射, 满足 $V' = \{v^{(1)}, v^{(2)} : v \in V\}$ 以及 $E' = \{\{u^{(1)}, v^{(2)}\}, \{u^{(2)}, v^{(1)}\} : \{u, v\} \in E\}$ 。在构造到 st-UCONN 的归约时, 注意: 顶点 v 位于图 G 的单圈中当且仅当 $v^{(1)}$ 和 $v^{(2)}$ 在 G' 中是连通的。在构造从 st-UCONN 到二部图的归约时, 注意: s 和 t 在 G 中以长为偶数 (或奇数) 的路径相连, 当且仅当向图 G' 增加一条边 $\{s^{(1)}, t^{(1)}\}$ 时, 它不再是二部图 (或增加边 $\{s^{(1)}, x\}$ 和 $\{x, t^{(2)}\}$, 其中 $x \notin V'$ 是一个辅助顶点)。

5.17 (寻找无向图中的路径) 续习题 5.15, 构造一个对数空间算法, 给定一个无向图 $G = (V, E)$ 及两个指定顶点 s 和 t , 求 G 中从 s 到 t 的路径 (如果该路径存在)。

提示: 续习题 5.15, 我们可以找到并 (暗中) 保存一条 $G_{O(\log|V|)}$ 中将代表 s 与代表 t 的顶点相连的对数长度路径。为了在 G_0 中找到一条对应于 $G_{O(\log|V|)}$ 中边的路径, 注意到可以将断言 5.9 与引理 5.10 相结合来构造归约, 再利用该归约找到路径 (参考文献 [190] 中给出了另一种描述)。

5.18 (NSPACE 两个模型间的关系) 参考 5.3.1 节中的定义, 证明对任意函数 s , 满足 $\log s$ 为限定空间可构造的, 且空间至少为对数, 则有 $N_{\text{SPACEon-line}}(s) = N_{\text{SPACEoff-line}}(\Theta(\log s))$ 。注意: 当 s 至少为对数时, $N_{\text{SPACEon-line}}(s) \subseteq N_{\text{SPACEoff-line}}(O(\log s))$ 也成立。

提示 (针对 $N_{\text{SPACEon-line}}(s) \subseteq N_{\text{SPACEoff-line}}(O(\log s))$): 利用离线机的非确定输入为在线机的一种可接受的计算编码, 即该输入中应该包含从初始配置到一种可接受配置之间所有的配置构成的序列, 而每个配置由工作带内容、机器状态以及机器在工作带和输入带上的位置构成。对离线机的模拟 (验证记录于非确定输入带上的配置序列的正确性) 只需保存当前检查的一对相继配置的位置, 从而需要的存储空间为单个配置的对数 (这又等于 $s(n) + \log_2 s(n) + \log_2 n + O(1)$) (注意: 验证过程依赖于对非确定的输入带的双向访问)。

提示 (针对 $N_{\text{SPACEoff-line}}(s') \subseteq N_{\text{SPACEon-line}}(\exp(s'))$): 这里涉及交叉序列的概念。具体地, 对于离线的非确定输入上的每个位置, 考虑机器的剩余配置序列, 其中每个剩余配置包含该

非确定带上的比特、机器的临时存储内容以及机器在输入带和存储带上的位置（但不包含机器在非确定带上的位置）。证明这样一个交叉序列的长度是离线机空间复杂度的指数，而离线机的时间复杂度至多为其空间复杂度的双指数（见习题 5.4）。在线机器只需（“在线地”）生成交叉序列，并检查交叉序列中每一对相继配置是一致的即可。这就要求保存两个交叉序列，需要的存储空间为序列长度的线性函数（是离线机空间复杂度的指数）。

5.19 (st-CONN 及其变形属于 NL) 证明以下计算问题属于 $\mathcal{NL}$ 类。问题的实例具有形式 (G, v, w, l) ，其中 $G=(V, E)$ 是一个有向图， $v, w \in V$ ， l 为整数，问 G 中是否包含从 v 到 w 的长度不超过 l 的路径。

提示：考虑一个能即时生成并验证路径的非确定机器。即从 $v_0 = v$ 开始，机器迭代运行，在第 i 次迭代中，非确定地生成 v_i ，验证是否 $(v_{i-1}, v_i) \in E$ ，并检查是否 $i \leq l$ 以及 $v_i = w$ 。注意：这样一个机器只需保存路径中最后两个顶点（即 v_{i-1} 和 v_i ）以及经过的边的个数（即 i ）（实际上，在一个谨慎的实现中，只需保存两个顶点之一（以及当前的 i 值）即可）。

5.20 (求有向图中的有向路径) 构造一个对数空间预言机，利用到 $\mathcal{NL}$ 的预言访问，能求出有向图中的（最短）有向路径。从而得出结论， $\mathcal{NL} = \mathcal{L}$ 当且仅当该路径可以利用（标准的）对数空间算法求得。

提示：利用到习题 5.19 中判定问题的归约，并根据习题 5.7 中的合成结论构造一个标准算法。

5.21 (NSPACE 和有向连通性) 目标是在通用的非确定空间受限计算与“强可构造 (Strongly Constructible)”图的有向连通性之间建立联系，其中图的规模是空间界限的指数。令 s 为限定空间构造的且至少是对数的函数。对每个 $S \in N_{\text{SPACE}}(s)$ ，构造一个线性时间预言机（有点类似于 5.2.4.2 节中的），在给定到 x 的预言访问时，能提供到规模为 $\exp(s(|x|))$ 的有向图 G_x 的预言访问，使得 $x \in S$ 当且仅当在 G_x 的第一个顶点与最后一个顶点之间存在一条有向路径。即对于一对输入 (u, v) 及 x 的预言访问，预言机可以确定 (u, v) 是否为 G_x 中的有向边。

提示：参考定理 5.11 的证明。

5.22 (定理 5.12 的另一种证明) 考虑每个顶点都有自环的有向图。

(1) 将有向图的邻接矩阵看作是预言（见习题 5.21），构造一个线性空间预言机，可以在输入图中判定一对给定结点是否由长为 2 的有向边相连。注意该预言机可以计算由预言所构造的图的平方图中的邻接关系。

(2) 利用简单合成（如引理 5.1），构造一个平方空间预言机，能在输入图中判定一对给定结点之间是否存在有向路径。

注意到，(2) 中的预言机蕴含了 st-CONN 可在对数平方空间内判定。特别地，证实了前面的自环假设是正确的。

5.23 (强连通性的判定) 一个有向图是强连通的，如果图中每对结点间存在一条有向路径（或等价地，任意一对结点之间都有有向回路通过）。证明，判定一个给定图是否为强连通图在（多到一的）对数空间归约下是一个 $\mathcal{NL}$ -完全问题。

提示 ($\mathcal{NL}$ -完全性)：将 st-CONN 归约为该问题。注意：对任意图 $G=(V, E)$ ， (G, s, t) 是 st-CONN 的一个“是”实例当且仅当图 $G'=(V, E \cup \{(v, s): v \in V\} \cup \{(t, v): v \in V\})$ 是强连通的。

5.24 (无向图中距离的确定) 证明以下计算问题在（多到一的）对数空间归约下是 $\mathcal{NL}$ -

完全的：给定一个无向图 $G=(V,E)$ ，两个指定结点 s 和 t ，以及整数 K ，确定 s 与 t 之间是否存在长度至多（或恰好）为 K 的路径。

提示 (NL-完全性)：将 st-CONN 归约为该问题。具体而言，给定一个有向图 $G=(V,E)$ 及结点 s, t ，考虑一个（“分层的”）图 $G'=(V',E')$ ，使得 $V'=\bigcup_{i=0}^{|V|-1}\{\langle i,v\rangle:v\in V\}$ 及 $E'=\bigcup_{i=0}^{|V|-2}\{\{\langle i,u\rangle,\langle i+1,v\rangle\}:(u,v)\in E\vee u=v\}$ 。注意： G 中存在从 s 到 t 的有向路径当且仅当 G' 中 $\langle 0,s\rangle$ 与 $\langle |V|-1,t\rangle$ 之间存在长度至多（或恰好）为 $|V|-1$ 的路径。

5.25 ($\mathcal{NL}\cap\text{co}\mathcal{NL}$, $\mathcal{NP}\cap\text{co}\mathcal{NP}$ 等，操作性的解释) 根据定义 5.13，证明 $S\in\mathcal{NL}\cap\text{co}\mathcal{NL}$ 当且仅当存在一个计算 χ_S 的非确定的对数空间机器，其中 $x\in S$ 时 $\chi_S(x)=1$ ，否则， $\chi_S(x)=0$ 。对于 $\mathcal{NP}\cap\text{co}\mathcal{NP}$ ，叙述并证明类似结论。

提示：从计算任意函数 f 的非确定机器出发，对每个值 v ，可以得到具有类似复杂度的非确定机器，能接受 $\{x:f(x)=v\}$ 。

（附加提示：调用计算 f 的机器 M ，并当且仅当 M 输出 v 时接受）。

另一方面，对任意值域有限的函数 f ，将其与接受各种集合 $S_v\stackrel{\text{def}}{=}\{x:f(x)=v\}$ 的非确定机器相结合，可以得到一个计算 f 的具有类似复杂度的非确定机器。

（附加提示：对于输入 x ，组合后的机器对 x 调用上述机器，并当且仅当接受 S_v 的机器接受才输出值 v 。如果这些机器都不接受，则组合后的机器输出 $\perp$ 。）

5.26 ($\mathcal{NL}=\text{co}\mathcal{NL}$ 的图论算法解释) 证明存在一个对数空间计算的函数 f ，对每个 (G,s,t) 而言， (G,s,t) 是 st-CONN 的一个“是”实例当且仅当 $(G',s',t')=f(G,s,t)$ 是 st-CONN 的一个“否”实例。

5.27 根据定义 5.13，证明存在一个非确定的对数空间机器，能计算给定无向图中两个给定结点间的距离。

提示：将该问题与习题 5.24 中的判定问题（的确切版本）相关联。

5.28 作为定理 5.14 证明中提出的双询问归约的替代，证明 st-CONN（特征函数的计算）可归约为在给定图中从一个给定结点出发，判定可达的结点个数，这里的归约是单询问的对数空间归约。

提示：对于输入 (G,s,t) ，其中 $G=(\{N\},E)$ ，考虑在图 $G'=(\{2N\},E\cup\{\langle t,N+i\rangle:i=1,\cdots,N\})$ 中从结点 s 出发，可达的结点数量。

5.29 (归约与非确定计算) 假设 f 的计算可在对数空间内归约为某个函数 g 的计算，且或者归约是非适应性的，或者对任意 x ，有 $|g(x)|=O(\log|x|)$ 。利用定义 5.13 中的非确定计算，证明如果存在一个计算 g 的非确定对数空间机器，则存在一个计算 f 的非确定对数空间机器。

提示：要点是将有关（计算 g 的）确定算法的组合结论应用于非确定的计算。具体地，在第一种情况下，利用习题 5.8 的结论，而在第二种情况下，利用习题 5.7 的结论。主要思想是：运行与确定情形相同的程序，并处理非确定机器在以自然方式计算 g 时可能出现的故障；即如果任何一个计算返回 $\perp$ ，则停止并输出 $\perp$ ，否则，与确定情形一样继续下去（利用得到的非 $\perp$ 的值）。

5.30 (QBF 游戏) 考虑以下由一个量化布尔公式开始的双方游戏。其中包含一个存在性参与方（试图证明公式是正确的）与一个全称性参与方（试图证明其不正确）。游戏过程如

下：参与方从左到右扫描公式，在遇到一个量词时，相应的参与方运行一步，其中包括对相应布尔变量的实例化。在最后一个位置，当所有的变量都被实例化之后，当且仅当相应的布尔表达式取值为真时，宣布获胜者为存在性参与方。

(1) 证明根据某些技术上的习惯，上述的 QBF 游戏适合高效双方游戏的框架（在 5.4 节开始描述的）。

(2) 证明任意高效的双方游戏可以表示为一个 QBF 游戏。

提示：对第 1 部分，在任何非最终位置（即并非所有变量都被实例化时），定义全称性参与方为获胜者；对第 2 部分，见第 3 章脚注。

第6章

随机性与计数

我将这种异乎寻常的多样性归功于一种制度：彩票。
那是别的国家所不知道，或者不够完善且不公开的。

豪尔赫·路易斯·博尔赫斯，巴比伦的彩票

迄今为止，我们研究计算设备的方法都有点保守：认为计算设备执行一套确定的规则。本章使用一个更自由和更贴近实际的方法，那就是认为计算设备使用的是一套概率规则。这种条件上的放宽直接影响了高效计算的概念，使其与概率多项式时间计算而不是确定的（多项式时间）计算相关。我们强调只有当计算的失效概率可以忽略不计时，高效计算与概率多项式时间计算的关系才有意义。

概率算法失效概率的量化特性在概率算法与计数问题之间建立起某种联系。后者实际上是一类新的计算问题，而我们关注的重点是对可高效识别对象的计数（如 NP 类集合的给定实例的 NP -证据）。在这类计数问题中随机化算法发挥着重要作用。

内容提要

重点讨论概率多项式时间算法，我们考虑这种算法各种类型的失效（如实际错误与无法产生输出的失效）。这需要定义几个复杂性类如 BPP 、 RP 和 ZPP 。本章的结论包括可以模仿概率多项式时间算法的多项式规模（非一致性）电路类的存在性（即 $BPP \subset P/poly$ ）以及 BPP 属于多项式时间层级（中的第二层）（即 $BPP \subseteq \Sigma_2$ ）。

然后介绍计数问题：具体来讲，就是对 PC 中搜索问题实例的解进行计数（或等价地，对 NP 中判定问题实例的 NP -证据计数）。我们会区分精确计数与近似计数（在相对近似的意义上）。特别地， PH 中任意问题可以归约到精确计数类 $\#P$ ，而（对 $\#P$ 的）近似计数可以（概率地）归约到 NP 。

通常，计数问题比相应的搜索（及判定）问题展现出“更丰富的结构”。例如，某些计数问题的精确版本是困难的（如 $\#P$ -完全问题），但其近似版本却是容易的，而其他一些计数问题的近似版本也是 NP -困难的。在某些情况下， $\#P$ -完全性来自于用以证明相应判定问题 NP -完全性的归约，而在其他情况下，则需要构造新的归约（通常是由于相应的判定问题并非 NP -完全，而是属于 P ）。

我们还考虑与近似计数相关的其他两类计算问题。第一类与承诺问题有关，称为唯一解

问题，其中问题的解决者保证实例至多有一个解。许多 NP-完全问题可以随机地归约到相应的唯一解问题。最后，讨论生成几乎均匀分布解的问题，并证明在许多情况下，该问题在计算上等价于对解的近似计数。

预备知识

假设读者熟悉概率论基础知识（见附录 D.1）。在 6.2 节，大部分内容都需要使用 2.1 节中给出的定义（即“搜索类 NP 问题” PC ，集合 $R(x) \stackrel{\text{def}}{=} \{y: (x, y) \in R\}$ ，以及 $S_R \stackrel{\text{def}}{=} \{x: R(x) \neq \emptyset\}$ ）。在 6.2.2 节~6.2.4 节，会大量涉及到附录 D.2 中的哈希函数及其性质。

6.1 概率多项式时间

考虑使用随机数的算法，我们将高效算法的概念由确定的多项式时间算法扩展到概率多项式时间算法。这样就出现了两个相互矛盾的问题，即允许随机化的计算步骤是否合理，以及增加这些步骤是否真的起作用。

首先要注意随机事件是我们这个世界的一个重要组成部分。这并不是说世界本身由真正的随机事件构成，而是指在建立模型时加入随机选项（即一些现象在某种意义上似乎是随机的）是有益的。另外，生成看似随机的事件（如掷硬币的结果）也是可行的^①。因此，假设计算机可以生成近似的随机数十分正常（实际上已经这样做了）。最后，这种假设会得到一种直观的计算模型，而研究这种模型也是出于自然的考虑。

这就导致一个问题，在计算模型中增加做出随机选择的功能是否有用。虽然人们认为随机化在一些计算环境中是必需的（如密码学（见附录 C）以及采样（见附录 D.3）），但问题在于解判定（及搜索）问题时，随机化是否有用。这确实是一个很好的问题，在 6.1.2.1 节中会进一步讨论。事实上，本节的一个主要目标就是提出该问题。为了阐述随机算法的优点，我们将给出几个例子（见 6.1.2.2 节、6.1.3.1 节和 6.1.5.2 节）。

6.1.1 基本模型

对基本计算模型进行扩展，可以形式化地定义概率（或随机）算法。我们以概率图灵机模型为例，但在这里（再次）强调对具体模型的选择并不重要（只要是“合理的”即可）。概率图灵机被定义为一种非确定的机器（见定义 2.7 中的第一项），但对其运算的定义却有本质上的不同。具体而言，定义 2.7 指的是该机器是否存在一种计算，（由某个给定输入开始）能到达某个配置，而在概率图灵机中，主要针对这样一个事件发生的概率，在计算的每一步，都需要在相关的可能选项中进行均匀选择。也就是说，如果机器的迁移函数将当前的状态—符号对映射为多个可能的三元组，则在相应的概率计算中，随机（等概地）选择其中的一个三元组并由此确定下一个配置。可以将这些随机选择看作是机器在内部掷硬币的结果（事实上，这就如同在非确定机器中，不失一般性，假设迁移函数将每个状态—符号对恰好映射为两个三元组，见习题 2.4）。

我们强调虚构的非确定机模型与概率机的现实模型之间存在本质区别。在非确定机模型

① 第 8 章及附录 D.4 中提出了关于随机化计算可行性的不同观点。第 8 章的重点是真正的随机性与表面随机性（对于计算受限的观察者而言）之间的区别。相反，附录 D.4 涉及到随机性的各种概念，以及将随机性的较弱形式转化为几乎完美形式的可行性。

中,考虑是否存在充分的选项序列(生成期望的输出),并忽略实际上如何做出选择。事实上,选择过程仅仅是一种智力实验。相反,在概率机模型中,每一步都要做出实际的随机选择(在一组预先确定的集合中均匀选择),并计算得到期望输出的概率。

根据上述讨论,我们要考虑这种概率机对于给定输入的输出分布,即对于概率机 M 及串 $x \in \{0,1\}^*$,用 $M(x)$ 表示对输入 x 调用 M 而产生的输出分布,其中概率取值来自于机器内部生成随机数中的均匀分布。毫无疑问,我们会考虑 $M(x)$ 为“正确”解的概率,即在搜索问题(或判定问题)情形中,重要的是 $M(x)$ 为实例 x 正确解(或表示 x 为正确判定)的概率。

上述讨论将机器内部生成随机数的过程看作是在线发生,即这些随机数是由机器自己以在线方式选择的。在另一种模型中,随机数序列由外部设备提供,保存于一个特殊的“随机输入”带上。此时认为随机数以离线方式生成。具体地,将给定了(初始)输入 x 和随机输入 r 之后,得到的确定机 M' 的输出记作 $M'(x,r)$ 。事实上, M' 是一个确定机,它有两个输入(第一个为实际输入,第二个为“随机输入”),但是我们考虑随机变量 $M(x) \stackrel{\text{def}}{=} M'(x, U_{l(|x|)})$, 其中 $l(|x|)$ 表示 $M'(x, \cdot)$ 所“期望的”随机数个数。

对于概率算法的这两种观点是紧密相关的:显然,从上述的剩余确定机 M' 出发,可以得到一个随机机器 M ,只需对输入 x ,随机选择一个具有足够长度的串 r ,并调用 $M'(x,r)$ 即可。另一方面,任意随机机器 M 的计算可以理解为是剩余机 M' 基于辅助输入 r (由 M' 生成,表示 M 内部生成的一个可能的随机数)对 $M(x)$ 的模拟(事实上,提供的随机数数量可以多于 M 的需要,这样不会有任何坏处,从而上述辅助输入的长度可以设置为与 M 的时间复杂度相等)。为了更清晰的阐述,并为以后的讨论提供参考,将上述讨论归纳为如下定义。

定义 6.1 (概率多项式时间的在线和离线定义)

- 称 M 是一个在线的概率多项式时间机,如果存在多项式 p ,使得对任意输入 $x \in \{0,1\}^*$,机器 M 在至多 $p(|x|)$ 步内停机(不考虑其内部随机数)。此时 $M(x)$ 为一个随机变量。

- 称 M' 为一个离线的概率多项式时间机,如果存在多项式 p ,使得对任意 $x \in \{0,1\}^*$ 及 $r \in \{0,1\}^{p(|x|)}$,在将 x 作为初始输入, r 作为随机输入时,机器 M' 在至多 $p(|x|)$ 步内停机。此时考虑的是随机变量 $M'(x, U_{p(|x|)})$, 其中 U_m 表示在 $\{0,1\}^m$ 上均匀分布的随机变量。

显然,对于时间复杂性,离线和在线定义是等价的(即给定一个在线的概率多项式时间机,可以得到一个功能完全相同的离线(概率多项式时间)机,反之亦然)。因此,后面我们将根据需要使用两个定义中更方便的一种。

我们强调,随机化算法的输出不再是其输入的函数,而更像是依赖于输入的一个随机变量。因此,解搜索问题和判定问题的定义(分别见定义 1.1 和定义 1.2)也要相应地调整。在调整时要考虑的主要问题是输出的值可能不一定正确。当然,我们希望输出正确的概率很大(但概率不一定是 1)。

错误概率

事实上,随机算法(概率机)的主要问题是它们可能会失效(见习题 6.1)。也就是说,这些算法以某种具体的(“失效”)概率无法产生期望的输出。我们讨论这种失效的两个方面:类型与量级。

(1) 失效的类型是一个定性概念。失效类型一方面要考虑当发生失效时, 算法是生成一个错误解, 还是仅仅指出无法找到正确解。另一方面, 对所有实例均失效还是仅对某种实例失效。为了澄清这些概念; 我们讨论 3 种类型的失效, 从而得到 3 种不同算法。

① 关于失效最宽松的概念是所谓双边错误(Two-sided Error)。这一术语来自于判定问题, 意思是(当失效时)算法可能在两个方向都出错(即可能将“是”实例判定为“否”实例, 或相反)。对于搜索问题, 双边错误的含义是: 算法在失效时, 会对任意输入输出错误解。也就是说, 算法可能会错误地判定输入无解, 也可能输出一个错误解(对有解的输入和无解的输入均是如此)。

② 中等程度的失效概念是所谓的单边错误(One-sided Error)。这个术语仍来自判定问题, 指算法只可能在一个方向上出错(即或者对“是”实例或者对“否”实例)。事实上, 根据算法对“是”实例出错或对“否”实例出错, 可自然地分为两种情况。在搜索问题中也有两种类似情况: 一种情况是算法从不输出错误解, 但会错误地判定输入无解; 另一种情况是不会误判输入无解, 但会输出一个错误解。

③ 最保守的失效概念是零边错误(Zero-sided Error)。此时, 算法的全部失误就是指出不能找到问题的解(通过输出一个特殊的“不知道”标记)。要强调此时算法从不提供错误解。

(2) 错误的量级是一个定量概念。它指算法失效的概率, 其中失效类型是固定的(如上述讨论)。

当实际使用一个随机算法时, 我们通常希望其失效概率可以忽略, 这意味着失效这一事件发生的概率极小, 以至于可以忽略。严格地说, 称一个量是可忽略的, 如果将其作为相关参数(如输入长度)的函数时, 这个量比任意正多项式的倒数衰减得更快。

为便于表达, 有时会考虑失效概率的其他上限。在选择这些上限时, 允许(并实际上促进了)“错误缩减”(即将满足该上限的概率多项式时间算法转化为一个失效概率可以忽略的算法)。例如, 在双边错误中, 要求能通过采样来区分正确解与错误解, 而在其他失效的类型中, 只需“命中”一个正确解即可。

以下 3 个小节(即 6.1.2 节~6.1.4 节)分别讨论对应于上述 3 种失效类型的复杂性类。为简单起见, 错误概率可以设为允许错误缩减的一个常量。

随机归约

在详细讨论之前, 我们指出随机归约在复杂性理论中起着重要作用。这种归约的定义类似于标准的 Cook-归约(或 Karp-归约), 但仍需考虑失效概率的类型和量级。为了表述清晰, 只写出双边错误版本。

- 类似于定义 2.9, 称问题 Π 可在概率多项式时间内归约为问题 Π' , 如果存在概率多项式时间预言机 M , 使得对任意解 Π' 的函数 f 以及任意 x , $M^f(x)$ 输出实例 x 的正确解的概率至少为 $1 - \mu(|x|)$, 其中 μ 是一个可忽略的函数。

- 类似于定义 2.11, 称判定问题 S 可通过随机化的 Karp-归约归约为判定问题 S' , 如果存在概率多项式时间算法 A , 使得对任意 x , 有 $\Pr[\chi_{S'}(A(x)) = \chi_S(x)] \geq 1 - \mu(|x|)$, 其中 χ_S (或 $\chi_{S'}$) 表示 S (或 S') 的特征函数, μ 为可忽略的函数。

这些归约是可以高效计算的, 并满足传递性, 见习题 6.2。

6.1.2 双边错误: BPP 类

本节讨论最自由但仍有意义的概率多项式时间算法。允许算法对每个输入出错, 但错误概率可忽略。这保证了算法是有用的, 因为在实际中, 非常小的错误概率确实可以忽略。

在重点讨论判定问题之前, 先对搜索问题进行一个简短的说明(见 1.2.2.1 节)。根据前面的定义, 称概率(多项式时间)算法 A 解关系 R 的搜索问题, 如果对任意 $x \in S_R$ (即 $R(x) \stackrel{\text{def}}{=} \{y: (x, y) \in R \neq \emptyset\}$), 有 $\Pr[A(x) \in R(x)] > 1 - \mu(|x|)$, 而对任意 $x \notin S_R$, 有 $\Pr[A(x) = \perp] > 1 - \mu(|x|)$, 其中 μ 为可忽略的函数。注意在这里并没有要求对有解的输入 x (即 $R(x) \neq \emptyset$) 调用算法时, 总是输出相同的解。事实上, 一个更强的要求是对任意有解的 x , 存在 $y \in R(x)$, 使得 $\Pr[A(x) = y] > 1 - \mu(|x|)$ 。这个条件及对其量化的放宽可以允许错误缩减(见习题 6.3)。

现在转向判定问题, 考虑错误概率可忽略的概率多项式时间算法。称一个概率(多项式时间)算法 A 能判定 S 的成员, 如果对任意 x , 有 $\Pr[A(x) = \chi_S] > 1 - \mu(|x|)$, 其中 χ_S 是 S 的特征函数(即当 $x \in S$ 时, $\chi_S(x) = 1$, 否则, $\chi_S(x) = 0$), μ 为可忽略的函数。可由概率多项式时间算法求解的判定问题类记作 **BPP**, 表示有限错误概率多项式时间(Bounded-error Probabilistic Polynomial-time)。实际上, 在标准定义中, 错误概率不超过 $1/3$ 。

定义 6.2 (BPP 类) 称判定问题 S 属于 **BPP** 类, 如果存在一个概率多项式时间算法 A , 使得对任意 $x \in S$, 有 $\Pr[A(x) = 1] \geq 2/3$, 而对任意 $x \notin S$, 有 $\Pr[A(x) = 0] \geq 2/3$ 。

这里选择常数为 $2/3$ 并不重要, 可以使用任意一个大于 $1/2$ 的常数(并得到相同的复杂性类)。类似地, 与其互补的常数 $1/3$ 也可用各种可忽略的函数代替(仍保持为同一类)。这两个事实都是关于 **BPP** 类健壮性的特殊情况, 以下利用错误缩减的过程来定义健壮性。

错误缩减(或信任放大)

对于映射 $\varepsilon: \mathbb{N} \rightarrow (0, 0.5)$, 令 BPP_ε 表示利用概率多项式时间算法可解, 且错误概率上限为 ε 的判定问题类, 即如果存在一个概率多项式时间算法 A , 使得对任意 x , 有 $\Pr[A(x) \neq \chi_S] \leq \varepsilon(|x|)$, 则 $S \in \text{BPP}_\varepsilon$ 。由定义可知, $\text{BPP} = \text{BPP}_{1/3}$ 。然而, 有更大范围的其他复杂性类也等于 **BPP**。特别地, 我们指出两个极端情况。

(1) 对任意正多项式 p 和 $\varepsilon(n) = (1/2) - (1/p(n))$, BPP_ε 类等于 **BPP** 类。即任意(“明显”)不同于 $1/2$ 的错误概率(即错误概率为 $(1/2) - (1/p(n))$) 都可缩减为 $1/3$ 。

(2) 对任意正多项式 p 及 $\varepsilon(n) = 2^{-p(n)}$, BPP_ε 类等于 **BPP** 类。即 $1/3$ 的错误概率可以进一步地缩减为指数衰减错误。

这两个事实都可以利用更弱的算法证明(即具有更大的错误概率上限), 只需运行次数足够多, 并根据多数原则来判定即可。我们强调在多次调用一个随机机器时, 不同调用中的随机选项是相互独立的。这个过程的成功概率可以利用大数原则(Law of Large Number)来分析(见习题 6.4)。

6.1.2.1 随机化的功能

现在转向 6.1 节一开始提出的问题, 即将高效计算的概念扩展到也包含概率多项式时间算法有何种作用。

这种说法似乎过于平淡。我们肯定会得到生成随机数的能力（并生成各种不同分布）。更具体地，随机化的算法在许多情况下是必需的（如可见第9章、10.1.2节、附录C及附录D.3），在另一些环境中也似乎是必需的（如可见6.2.2节~6.2.4节）。这里要问的是在解判定问题和搜索问题的受限情形时，随机化是否能增加多项式时间算法的计算能力。

问题在于， BPP 类是否能被扩展得超出 P 类（这里显然有 $P \subseteq BPP$ ）。通常假设答案是否定的。特别地，在某些合理假设下，有 $BPP = P$ （见定理8.19的第一部分）。然而，要注意在证明后一个结论时，会产生多项式级的减速，即运行于 $t(\cdot)$ 时间的随机算法可以利用运行于 $\text{poly}(t(\cdot))$ 时间的确定算法来模拟。这种减速是上述方法的固有缺陷（见8.3.3.2节）。进一步地，对某些具体问题（最有名的是素性检测（见6.1.2.2节）），已知的概率多项式时间算法明显比确定的多项式时间算法速度更快（并且也更简单）。因此，我们认为即使是在判定问题中，概率多项式时间算法也是有优势的。

注意即便是随机化不能提供任何计算上的优势（也就是说，即使每个能在概率多项式时间内判定的问题也都能用复杂性相同的确定算法判定时）， BPP 类的基本特性也保持不变。这一结论将从根本上回答随机性的能力问题^①。现在我们由上述的哲学性质讨论（且部分为假设）转向对 BPP 类的具体讨论。

BPP属于多项式时间层级

虽然 $BPP = P$ 可能成立，但 BPP 类是否包含于 NP 中尚且未知。原因在于 BPP 中的双边错误概率与 NP 定义中完全拒绝“否”实例的要求互不兼容（见习题6.8）。鉴于这种未知， BPP 包含于多项式时间层级的第二层（即 $BPP \subseteq \Sigma_2$ ）是很意义的。这是定理6.9的一个推论。

平凡的去随机化

消除算法随机性的一种直接方法是对所有内部随机数都尝试运行一遍，收集相关的统计数据并做出相应判断。这表示 $BPP \subseteq PSPACE \subseteq EXP$ ，而后两者常被看作是对 BPP 类进行平凡去随机后的结果。在8.3节，我们将考虑对 BPP 类各种非平凡的去随机化，这建立于各种困难性假设之上。读者也许不清楚去随机化与计算困难性之间的联系，这可以参考第8章。

非一致的去随机化

在许多环境中（特别是在解搜索问题及判定问题时），随机性的功能由非一致的建议所取代。直观上，非一致的建议是指规定了一个随机数序列，可以作为具有固定长度的（初始）输入。无论是解搜索还是判定问题，这种建议都能当成输入，^②而其存在性仅由足够低的错误概率（从而支持这些概率的联合上限）来保证。后者又可由错误缩减保证，因此有如下结论。

定理6.3 BPP 是 P/poly 的真子集。

证明：注意： P/poly 中包含了不可判定的问题（定理3.7），这些问题肯定不属于 BPP 。因此，我们主要证明 $BPP \subseteq P/\text{poly}$ 。由对错误缩减的讨论，对任意 $S \in BPP$ ，存在一个（确

① 类似地，证明 $IP = PSPACE$ （见定理9.4）并不会削弱这两类中任一类的重要性，因为每一类都代表着根本不同的模型。

② 在其他环境中（如可见第7章和第8章），有一个平均情况下是良好的建议就足够了，这里的平均是指在所有相关的（初始）输入中。

定性的) 多项式时间算法 A 和多项式 p , 使得对任意 x , 有 $\Pr\left[A\left(x, U_{p(|x|)}\right) \neq \chi_S(x)\right] < 2^{-|x|}$ 。利用联合上限, 有 $\Pr_{r \in \{0,1\}^{p(n)}}\left[\exists x \in \{0,1\}^n \text{ 使得 } A(x, r) \neq \chi_S(x)\right] < 1$ 。因此, 对任意 $n \in \mathbb{N}$, 存在串 $r_n \in \{0,1\}^{p(n)}$ 使得对任意 $x \in \{0,1\}^n$, 有 $A(x, r_n) = \chi_S(x)$ 。利用这样的序列 r_n 作为建议, 可以得到需要的非一致性机器 (这就证明了 $S \in P/\text{poly}$)。■

思考: 定理 6.3 的证明结合了错误缩减与概率方法 (见参考文献[11]) 的简单应用, 其中的概率方法通过分析随机对象的概率来证明其存在性。在这种情况下, 要寻找一个非一致的建议, 并通过分析一个随机建议为良好建议的概率来证明其存在性。在分析时, 采用的方法是在随机算法可能的内部随机数序列集合中识别可能的建议空间。

6.1.2.2 概率多项式时间素性检测

教学注释: 虽然最近证明了素性检测问题属于 P 类, 但我们相信以下的例子很好地说明了随机算法的能力。

我们提出一个简单的概率多项式时间算法, 用于判定一个给定整数是否为素数。这里用到的数论知识仅仅是以下两个事实。

事实 1: 对任意素数 $p > 2$, $\text{mod } p$ 的每个平方剩余恰有两个 $\text{mod } p$ 的平方根 (且其和为 p)^①。

事实 2: 对任意 (奇数且非整数幂的) 合数 N , $\text{mod } N$ 的每个平方剩余至少有 4 个 $\text{mod } N$ 的平方根。

我们提出的算法使用另一个算法作为黑盒, 记作 sqrt , 该算法给定素数 p 和 $\text{mod } p$ 的平方剩余, 记作 s , 能返回 $s \text{ mod } p$ 的两个平方根中较小的一个。当然, 如果输入不满足上述条件, 则不保证输出也符合要求 (特别是当 p 不是素数时)。因此, 实际上给出了一个概率多项式时间归约, 将素性检测归约到提取模素数的平方根问题 (这是一个有意义的搜索问题, 见 2.4.1 节)。

构造 6.4 (素性检测到求模素数平方根的归约) 对于输入的自然数 $N > 2$:

- (1) 如果 N 为偶数或整数的平方, ^②则拒绝;
- (2) 均匀选择 $r \in \{1, 2, \dots, N-1\}$, 并令 $s \leftarrow r^2 \text{ mod } N$;
- (3) 令 $r' \leftarrow \text{sqrt}(s, N)$, 如果 $r' \equiv \pm r \pmod{N}$, 则接受, 否则拒绝。

事实上, 当 N 为合数时, 归约中是对非法输入调用 sqrt 函数 (即发出的询问与目标问题中的承诺相悖)。此时, 不能保证 sqrt 会返回什么值, 但是显然会得到错误的应答。通常我们会证明如果 N 为合数, 则无论 sqrt 返回何值, 归约拒绝的概率至少为 $1/2$ 。还要指出确实存在实现 sqrt 函数的概率多项式时间算法 (见习题 6.16)。

命题 6.5 构造 6.4 中构造了由素性检测到求模素数平方根问题的概率多项式时间归约。进一步地, 如果输入为素数, 则归约总是接受, 否则, 拒绝的概率至少为 $1/2$ 。

我们强调命题 6.5 针对的是归约本身, 即 sqrt 函数被看作是一个 (“完善的”) 预言: 对任意素数 P 及 $\text{mod } P$ 的平方剩余 s , 能返回 $r < s/2$ 使得 $r^2 \equiv s \text{ mod } P$ 。将命题 6.5 与计算 sqrt

① 即对任意 $r \in \{1, 2, \dots, p-1\}$, 方程 $x^2 \equiv r^2 \text{ mod } p$ 有两个根 (即 r 和 $p-r$)。

② 检查方法是, 扫描 $e \in \{2, \dots, \log_2 N\}$ 的所有幂, 并且对 e 的所有取值, (近似地) 解方程 $x^e = N$ (即寻找最小整数 i 使得 $i^e \geq N$)。这个解可以用二分查找法求得。

的概率多项式时间算法（其错误概率可忽略）相结合，可以证明素性检测问题属于 BPP 。

证明：由事实 1，对于输入的素数 N ，构造 6.4 总是接受（因为此时对任意 $r \in \{1, 2, \dots, N-1\}$ ，有 $\text{sqrt}(r^2 \bmod N, N) \in \{r, N-r\}$ ）。另一方面，假设 N 为奇合数且不是整数的幂，则由事实 2 可知，每个平方剩余 s 都至少有 4 个平方根，并且这些根在步骤 (2) 中被选中的概率相等（换句话说，从 s 中得不到步骤 (2) 中所选平方根的任何信息）。因此，对任意这样的 s ， $\text{sqrt}(s, N)$ 或 $N - \text{sqrt}(s, N)$ 与步骤 (2) 中选择的平方根相等的概率至多为 $2/4$ 。从而，当输入为合数时，归约拒绝的概率至少为 $1/2$ 。■

思考：构造 6.4 阐明了随机算法（或归约）一个有趣的特点，即可以利用调用的子程序中的未知信息。具体而言，构造 6.4 生成了一个算法实例 (N, s) ，其中隐藏了关键信息（关于 s 是如何生成的）。当 N 为素数时，任意能做出正确回答的子程序都提供了 N 是素数的概率证据，而概率空间中也包括丢失的信息（ N 为合数时，关于如何生成 s 的信息）。

说明：素性检测问题实际上属于 P 类，然而，解决这一问题的确定算法远比构造 6.4 复杂得多（分析起来则更复杂）。

6.1.3 单边错误：RP 和 coRP 类

本节讨论具有单边错误的概率多项式时间算法。单边错误的概念涉及到对问题实例集的自然划分，即判定问题中的“是”实例与“否”实例，以及搜索问题中有解的实例与无解的实例。我们主要考虑判定问题，并说明对搜索问题的类似的处理方式（见习题 6.3）。

定义 6.6 (RP 类)^① 设 S 为判定问题，如果存在一个概率多项式时间算法 A ，使得对任意 $x \in S$ ，有 $\Pr[A(x)=1] \geq 1/2$ ，而对任意 $x \notin S$ ，有 $\Pr[A(x)=0]=1$ ，则称 S 属于 RP 。

这里选择常数为 $1/2$ 并非实质性的，可以使用任意大于 0 的常数（并得到同一个类）。类似地，该常数可以用 $1 - \mu(|x|)$ 代替，其中 μ 为可忽略的函数（仍得到同一个类）。这两个事实都表明了 RP 类的健壮性（见习题 6.5）。

观察到 $RP \subseteq NP$ （见习题 6.8）以及 $RP \subseteq BPP$ （由上述的错误缩减）。定义 $coRP = \{0, 1\}^* \setminus S : S \in RP\}$ ，注意： $coRP$ 对应着单边错误概率的反面。即判定问题 S 属于 $coRP$ ，如果存在概率多项式时间算法 A ，使得对任意 $x \in S$ ，有 $\Pr[A(x)=1]=1$ 而对任意 $x \notin S$ ，有 $\Pr[A(x)=0] \geq 1/2$ 。

6.1.3.1 多项式相等的测试

单边错误随机算法的一个有趣例子是判定两个多项式是否相等。为简单起见，假设给定了对两个多项式求值的预言机。或者采用另一种表述，以算术电路形式表示多项式（参考附录 B.3）可得到 $coRP$ 中一个标准的判定问题（见习题 6.17）。无论采用哪种方法，我们讨论的都是多变量多项式，判定其在任意域中是否相同（或等价地，在一个足够大的有限域中是否相同）。注意只要考虑比两个多项式的次数都大的有限域即可。

构造 6.7（对多项式是否相同的检测） 设 n 为整数， F 为有限域，给定对于 $p, q: F^n \rightarrow F$ 的黑盒访问，均匀选择 $r_1, r_2, \dots, r_n \in F$ ，当且仅当 $p(r_1, r_2, \dots, r_n) = q(r_1, r_2, \dots, r_n)$ 时接受。

① RP 的原始定义表示随机多项式时间（Random Polynomial time），其中并不反映该类中对错误类型进行限制的事实。这一概念唯一的优点是使人联想到 NP ，从而体现出一个事实，即 RP 是 NP 中一个随机化的多项式时间类。

显然, 如果 $p \equiv q$, 则构造 6.7 中的算法总是接受。以下引理表明, 如果 p 和 q 为有限域 F 上不同的多项式, 并且次数均不大于 d , 则上述算法接受的概率最多为 $d/|F|$ 。

引理 6.8 设 $p: F^n \rightarrow F$ 为有限域 F 上的 d 次非零多项式, 则有

$$\Pr_{r_1, r_2, \dots, r_n \in F} [p(r_1, r_2, \dots, r_n) = 0] \leq \frac{d}{|F|}$$

证明: 对 n 进行归纳。当 $n=1$ 时, 由代数中的基础理论直接可证 (即 d 次非零单变量多项式最多有 d 个不同根)。在归纳步骤中, 将 p 写为关于其第一个变量的多项式, 即

$$p(x_1, x_2, \dots, x_n) = \sum_{i=1}^d p_i(x_2, x_3, \dots, x_n) \cdot x_1^i$$

式中: p_i 为次数不超过 $d-i$ 的多项式。令 i 为满足使每个 p_i 都不为零的最大整数, 去掉 $i=0$ 的情形并利用归纳假设, 有

$$\begin{aligned} & \Pr_{r_1, r_2, \dots, r_n} [p(r_1, r_2, \dots, r_n) = 0] \\ & \leq \Pr_{r_2, r_3, \dots, r_n} [p_i(r_2, r_3, \dots, r_n) = 0] + \Pr_{r_1, r_2, \dots, r_n} [p(r_1, r_2, \dots, r_n) = 0 \mid p_i(r_2, r_3, \dots, r_n) \neq 0] \\ & \leq \frac{d-i}{|F|} + \frac{i}{|F|} \end{aligned}$$

其中对第二项求最大值时, 固定任意使 $p_i(r_2, r_3, \dots, r_n) \neq 0$ 的序列 $r_2, r_3, \dots, r_n$, 并考虑单变量多项式 $p'(x) \stackrel{\text{def}}{=} p(x, r_2, \dots, r_n)$ (由假设可知, 这是一个 i 次非零多项式)。

思考: 可以认为, 引理 6.8 的含义是对 F 上任意的 d 次非零多项式, 其定义域中占 $1-(d/|F|)$ 比例的部分, 求值不为零。因此, 如果 $d \ll |F|$, 则定义域中大部分点都是多项式取值非零的证据。尚不知道是否存在高效的确定性算法, 给定一个由算术电路表示的多项式, 能找到这样的证据。事实上, 构造 6.7 仅通过随机选择来寻找证据。

6.1.3.2 BPP 与 RP 的关系

关于概率多项式时间算法的一个自然问题是双边错误与单边错误概率间的关系。例如, 是否 BPP 包含于 RP 中? 不严格地说, 可以通过单边错误的随机 Karp-归约来证明 BPP 可以归约为 $coRP$, 其中使用了这两个类中的承诺问题 (下一段中给出简要定义)。注意: 如果使用双边错误的随机 Karp-归约, 则可将 BPP 平凡地归约为 $coRP$, 而 BPP 到 $coRP$ 的确定 Karp-归约蕴含了 $BPP = coRP = RP$ (见习题 6.9)。

首先, 建议读者参考 2.4.1 节对承诺问题的一般性介绍。类似于定义 2.31, 称承诺问题 $\Pi = (S_{\text{yes}}, S_{\text{no}})$ 属于 BPP (的承诺问题扩展), 如果存在概率多项式时间算法 A , 使得对任意 $x \in S_{\text{yes}}$, 有 $\Pr[A(x)=1] \geq 2/3$, 而对任意 $x \in S_{\text{no}}$, 有 $\Pr[A(x)=0] \geq 2/3$ 。类似地, 称 Π 属于 $coRP$, 如果对任意 $x \in S_{\text{yes}}$, 有 $\Pr[A(x)=1]=1$, 而对任意 $x \in S_{\text{no}}$, 有 $\Pr[A(x)=0] \geq 1/2$ 。承诺问题之间概率归约的定义符合 2.4.1 节中的习惯, 具体而言, 对于违背归约目标中承诺的询问, 应答是随意给出的。

定理 6.9 BPP 中任意问题可以通过单边错误的随机 Karp-归约归约到 $coRP$, 其中 $coRP$ (也可能是 BPP) 表示相应的承诺问题类。特别地, 归约过程总是将“否”实例映射为“否”实例。

从而, BPP 可由单边错误的随机 Cook-归约归约为 RP 。因此, 利用 3.2.2 节中的习惯,

并考虑承诺问题类, 可以写成 $BPP \subseteq RP^{RP}$ 。事实上, 由于 $RP^{RP} \subseteq BPP^{BPP} = BPP$, 从而有 $BPP = RP^{RP}$ 。定理 6.9 可以解释为将归约的单边错误概率与 $coRP$ 的单边错误概率联合起来, 就构成了 BPP 的双边错误概率。要注意的是, 这并不像 $1+1=2$ 那样显然, 特别是我们还不知道在标准的判定问题类 (而非定理 6.9 中所考虑的承诺问题类) 中定理是否成立。

证明: 回顾 BPP 算法的错误概率是易缩减的, 并能得到错误概率呈指数衰减的概率多项式算法。但是这并不会彻底消灭错误 (即使是在“一边”)。一般来说, 没有可能完全消除错误, 除非 (或者做出某种惊天动地的事情, 或者) 改变计算环境, 允许单边错误随机归约到 $coRP$ 中的某个问题。改变后的计算环境可以看作是由两个步骤构成的随机游戏 (一个是归约, 另一个是 $coRP$ 中的判定程序), 并允许对两个步骤使用不同的量词 (即第一个量词允许一个方向上的错误, 而第二个量词允许另一个方向上的错误)。下一段将阐述这种环境的功能, 这对真正的证明来说至关重要。

教学注释: 以下阐述定理 6.9 的另一种证明方法。这种方法似乎在概念上更简单, 但是需要一个起点 (或一个假设), 而这样的起点很难建立。其中的两种比较都与定理 6.9 的实际证明有关。

一个例子

假设对某个集合 $S \in BPP$, 存在多项式 p' 和一个离线的 BPP-算法 A' , 使得对任意 x , 有 $\Pr_{r \in \{0,1\}^{2p'(|x|)}} [A'(x,r) \neq \chi_S(x)] < 2^{-(p'(|x|)+1)}$, 也就是说, 算法使用 $2p'(|x|)$ 比特的随机数, 并且错误概率小于 $2^{-p'(|x|)}/2$ 。注意: 这样一个算法不能由标准的错误缩减 (见习题 6.10) 得出。无论如何, 这么小的错误概率允许对串 r 进行分割, 使其中的一段对“是”实例的全部错误概率负责, 而另一段对“否”实例的全部错误概率负责。具体来说, 对任意的 $x \in S$, 有 $\Pr_{r' \in \{0,1\}^{p'(|x|)}} \left[\left(\forall r'' \in \{0,1\}^{p'(|x|)} \right) A(x, r'r'') = 1 \right] > 1/2$, 而对任意 $x \notin S$ 及 $r' \in \{0,1\}^{p'(|x|)}$, 有 $\Pr_{r'' \in \{0,1\}^{p'(|x|)}} [A'(x, r'r'') = 1] < 1/2$ 。从而“是”实例的错误被“推”到了与 r' 的选择相关, 而“否”实例的错误被“推”到了与 r'' 的选择相关。这样就得到了一个单边错误的随机 Karp-归约, 将实例 x 映射为 (x, r') , 其中 r' 从 $\{0,1\}^{p'(|x|)}$ 中均匀选择, 从而将判定问题 S 归约为 $coRP$ 问题 (考虑对 (x, r')), 可由 (在线的) 随机算法 A'' 判定, 其中 A'' 定义为 $A''(x, r') \stackrel{\text{def}}{=} A'(x, r'U_{p'(|x|)})$ 。细节详见习题 6.11。以下给出实际证明, 其中避免了上述假设。

实际出发点: 考虑任意一个特征函数为 χ 的 BPP 问题 (在承诺问题中, χ 为部分函数, 只对承诺有定义)。由标准的错误缩减, 存在一个概率多项式时间算法 A , 使得对任意使 χ 有定义的 x , 有 $\Pr[A(x) \neq \chi(x)] < \mu(|x|)$, 其中 μ 为可忽略的函数。在相应的剩余 (离线) 算法 A' 中, 将 A 的运行时间界限表示为多项式 p , 则对所有使 χ 有定义且足够长的 x , 有

$$\Pr_{r \in \{0,1\}^{p(|x|)}} [A'(x,r) \neq \chi(x)] < \mu(|x|) < \frac{1}{2p(|x|)} \quad (6.1)$$

我们将构造一个随机的单边错误 Karp-归约, 将 χ 归约为 $coRP$ 中的承诺问题。

教学注释: 某些读者可能更愿意跳过以下两段, 直接学习后面关于随机映射的形式化描述。对于这些读者, 我们建议在阅读了形式化分析之后再回来阅读跳过的两段。

主要思想：上述例子的基本思想是将 (χ) 的“是”实例中的错误概率“集中”到归约中，而将“否”实例的错误概率集中到 coRP 问题。主要讨论 $\chi(x)=1$ 的情形，为达到这个目的，需要在输入 x 中增加一个随机的“修饰”序列，作用于算法 A' 的随机输入，使得当修饰的选择恰当时，对任意 $r \in \{0,1\}^{p(|x|)}$ ，序列中存在一个修饰，作用于 r 之后得到 r' ，满足 $A'(x, r')=1$ 。事实上，并非所有的修饰序列都是恰当的，但是一个随机序列满足条件的概率很高，而不满足上式的序列可归入归约中的错误。另一方面，如果只使用排列方式不同的修饰，则保证了“否”实例的错误概率只增加一个与修饰数量相当的因子，而这个错误概率也可归入 coRP 问题的错误概率。具体细节如下。

上述的修饰可由移位（即带固定偏移量的所有串的集合）来实现。因此，向输入的 x 中增加一个随机的移位序列，记作 $s_1, s_2, \dots, s_m \in \{0,1\}^{p(|x|)}$ ，使得对于恰当选择的 $(s_1, s_2, \dots, s_m)$ ，对任意 $r \in \{0,1\}^{p(|x|)}$ ，存在 $i \in [m]$ ，使得 $A'(x, r \oplus s_i)=1$ 。我们将证明，对任意的“是”实例 x 和适当选择的 m ，随机选择的移位序列为恰当序列的概率极高。因此，对 $A''(\langle x, s_1, \dots, s_m \rangle, r) \stackrel{\text{def}}{=} \bigvee_{i=1}^m A'(x, r \oplus s_i)$ ，相应于 $s_1, s_2, \dots, s_m$ 的选择而言，将“是”实例 x 映射为扩张后的、被 A'' 以概率 1 接受的输入 $\langle x, s_1, \dots, s_m \rangle$ 的概率极高。另一方面，对于扩张后的“否”实例（其中的移位任意选择），接受概率只增加了 m 倍。为了进一步详细描述上述思想，我们从简单的随机映射（作为随机 Karp-归约）开始，然后定义目标承诺问题。

随机映射：对于输入 $x \in \{0,1\}^n$ ，令 $m = p(|x|)$ ，均匀选择 $s_1, s_2, \dots, s_m \in \{0,1\}^m$ ，并输出对 $(x, \bar{s})$ ，其中 $\bar{s} = (s_1, s_2, \dots, s_m)$ 。注意：这个映射，记作 M ，可由概率多项式时间算法计算。

承诺问题：定义如下的承诺问题 $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ ，问题的实例具有形式 $(x, \bar{s})$ ，其中 $|\bar{s}| = p(|x|)^2$ 。

- 对 $(x, \bar{s})$ 为“是”实例，如果对任意 $r \in \{0,1\}^m$ ，存在 i 满足 $A'(x, r \oplus s_i)=1$ ，其中 $\bar{s} = (s_1, s_2, \dots, s_m)$ ，且 $m = p(|x|)$ 。
- 对 $(x, \bar{s})$ 为“否”实例，如果对任意 i ，至少有一半的 $r \in \{0,1\}^m$ 使得 $A'(x, r \oplus s_i)=0$ ，其中 $\bar{s} = (s_1, s_2, \dots, s_m)$ ，且 $m = p(|x|)$ 。

为了说明 Π 确实是一个 coRP 类承诺问题，考虑如下的随机算法：对于输入 $(x, (s_1, s_2, \dots, s_m))$ ，其中 $m = p(|x|) = |s_1| = \dots = |s_m|$ ，算法均匀选择 $r \in \{0,1\}^m$ ，当且仅当对某个 $i \in \{1, 2, \dots, m\}$ ， $A'(x, r \oplus s_i)=1$ 时，接受该输入。事实上， Π 的“是”实例被接受的概率为 1，而“否”实例被拒绝的概率至少为 $1/2$ 。

对归约的分析：我们断言具有单边错误的随机映射 M 将 χ 映射为 Π ，具体地，要证明两个断言。

断言 1：如果 x 为“是”实例（即 $\chi(x)=1$ ），则 $\Pr[M(x) \in \Pi_{\text{yes}}] > 1/2$ 。

断言 2：如果 x 为“否”实例（即 $\chi(x)=0$ ），则 $\Pr[M(x) \in \Pi_{\text{no}}] = 1$ 。

先证明较容易的断言 2。回顾 $M(x) = (x, (s_1, s_2, \dots, s_m))$ ，其中 $s_1, s_2, \dots, s_m$ 在 $\{0,1\}^m$ 中均匀、独立分布。注意：（由式 (6.1) 及 $\chi(x)=0$ ）对所有可能的 $s_1, s_2, \dots, s_m \in \{0,1\}^m$ 及所有

$i \in \{1, 2, \dots, m\}$, r 满足 $A'(x, r \oplus s_i) = 1$ 的概率至多为 $\frac{1}{2m}$ 。因此, 对任意可能的

$s_1, s_2, \dots, s_m \in \{0, 1\}^m$ 及至多一半的 $r \in \{0, 1\}^m$, 存在一个 i , 使得 $A'(x, r \oplus s_i) = 1$ 成立。因此, 归约 M 总是将“否”实例 x (即 $\chi(x) = 0$) 映射到 Π 的“否”实例 (即 Π_{no} 中元素)。

现在证明断言 1 (针对 $\chi(x) = 1$ 的情形), 我们将扼要地证明, 此时, 归约 M 以极高的概率将 x 映射为 Π 的“是”实例。计算归约的失效 (当 $\chi(x) = 1$ 时) 概率上限如下:

$$\begin{aligned} \Pr[M(x) \notin \Pi_{\text{yes}}] &= \Pr_{s_1, s_2, \dots, s_m} [\exists r \in \{0, 1\}^m, \text{使得} (\forall i) A'(x, r \oplus s_i) = 0] \\ &\leq \sum_{r \in \{0, 1\}^m} \Pr_{s_1, s_2, \dots, s_m} [(\forall i) A'(x, r \oplus s_i) = 0] \\ &= \sum_{r \in \{0, 1\}^m} \prod_{i=1}^m \Pr_{s_i} [A'(x, r \oplus s_i) = 0] \\ &< 2^m \cdot \left(\frac{1}{2m}\right)^m \end{aligned}$$

其中最后一个不等式可由式 (6.1) 推出。由上式可知, 如果 $\chi(x) = 1$, 则 $\Pr[M(x) \in \Pi_{\text{yes}}] \gg 1/2$ 。

将两个断言相结合, 可以得到随机映射 M 将 χ 映射为 Π , 且关于“是”实例存在单边错误, 再联想到 $\Pi \in \text{coRP}$, 定理得证。■

BPP 包含于 PH

人们习惯上利用定理 6.9 的证明思想来证明 BPP 包含于多项式时间层级中 (这两个类中都针对着其中的标准判定问题)。具体地, 为证明 $\text{BPP} \subseteq \Sigma_2$ (见定义 3.8), 定义多项式时间计算的谓词 $\varphi(x, \bar{s}, r) \stackrel{\text{def}}{=} \bigvee_{i=1}^m (A'(x, s_i \oplus r) = 1)$, 并观察到

$$\chi(x) = 1 \Rightarrow \exists \bar{s} \forall r \varphi(x, \bar{s}, r) \quad (6.2)$$

$$\chi(x) = 0 \Rightarrow \forall \bar{s} \exists r \neg \varphi(x, \bar{s}, r) \quad (6.3)$$

(其中式 (6.3) 等价于 $\neg \exists \bar{s} \forall r \varphi(x, \bar{s}, r)$)。注意: 断言 1 (定理 6.9 的证明中) 中指出大部分序列 $\bar{s}$ 满足 $\forall r \varphi(x, \bar{s}, r)$, 而式 (6.2) 只要求至少存在一个这样的 $\bar{s}$ 。类似地, 断言 2 中指出对任意 $\bar{s}$, 大部分 r 都使 $\varphi(x, \bar{s}, r)$ 不成立, 而式 (6.3) 只要求对任意 $\bar{s}$, 至少存在一个这样的 r 。我们认为利用同样的证明思想可以得到类似结论 (如 $\text{BPP} \subseteq \text{MA}$, 其中 MA 为 NP 的随机化版本, 定义见 9.1 节)。^①

6.1.4 零边错误: ZPP 类

现在考虑从不出错, 但可能无法提供答案的概率多项式时间算法。主要讨论判定问题, 相应的类记作 ZPP (表示零错误概率多项式时间)。 ZPP 的标准定义中使用了以至多 $1/2$ 的

① 具体地, 在定义 MA 类时, 允许定义 2.5 中的验证算法为概率算法, 并允许其对“否”实例出错。也就是说, 对任意 $x \in S$, 存在 $y \in \{0, 1\}^{\text{poly}(|x|)}$ 使得 $\Pr[V(x, y) = 1] = 1$, 而对任意 $x \notin S$ 及所有 y , 有 $\Pr[V(x, y) = 0] \geq 1/2$ 。注意: MA 可以看作是上述两个条件的混合, 具体地, MA 中每个问题必须同时满足式 (6.2) 和断言 2。习题 6.12 介绍了 NP (即 MA 的变形) 的其他随机化版本。

概率输出 $\perp$ (指示失效) 的机器。称 $S \in ZPP$, 如果存在概率多项式时间算法 A , 使得对任意 $x \in \{0,1\}^*$, 有 $\Pr[A(x) \in \{\chi_S(x), \perp\}] = 1$, 且 $\Pr[A(x) = \chi_S(x)] \geq 1/2$, 其中当 $x \in S$ 时, $\chi_S(x) = 1$, 否则 $\chi_S(x) = 0$ 。同样, 这里常量 (即 $1/2$) 的选择并不重要, 而且可以用“错误缩减”证明, 能以显著优势得到有意义回答的算法可以被放大为失效概率可忽略的算法 (见习题 6.6)。

定理 6.10 $ZPP = RP \cap coRP$ 。

证明概要: 可以对 ZPP 算法进行平凡转化来证明 $ZPP \subseteq RP$ (以及 $ZPP \subseteq coRP$), 也就是说, 用一个“否”判决 (或“是”判决) 来代替错误指示符 $\perp$ 。注意: 当 ZPP 算法失效时如何表示取决于允许的错误类型。

为证明 $RP \cap coRP \subseteq ZPP$, 将 $RP \cap coRP$ 中一个集合对应的两个算法相结合。要点的在于当回答为“是” (或“否”) 时可以信任 RP -算法 (或 $coNP$ -算法), 但是当回答为“否” (或“是”) 时并不一定信任。因此, 同时调用这两个算法, 并且仅当得到可以信任的回答 (以很高概率出现) 时才输一个出明确的答案, 否则, 输出 $\perp$ 。 $\square$

期望的多项式时间

对于某些资源, ZPP 用运行于期望的多项式时间的随机算法来定义, 并总是输出正确答案。这个定义等价于我们使用的定义 (见习题 6.7)。

6.1.5 随机的对数空间

本节讨论具有更多限制的随机多项式时间算法, 这些算法只允许使用对数空间。

预备知识

从技术上看, 本小节是自包含的。然而, 感兴趣的读者可以参考第 5 章中关于空间复杂性的更多内容。

6.1.5.1 定义

在定义空间受限的随机算法时, 我们面临的问题与非确定的空间受限计算 (见 5.3 节) 中的问题相类似。具体地, 在线和离线版本 (见定义 6.1) 不再是等价的, 除非限制“离线机”只能以单向方式访问其输入带。问题在于, 在空间受限计算中 (与只考虑时间上限的情况不同), 记录内部产生的随机数 (在线模式下) 要付出代价。要记住的是, 在这里我们希望能对真正的算法建立模型 (而不是像 5.3 节一样提出一个假想的模型, 只用于解释一种基本现象), 显然, 使用在线版本是正常的选择。

另一个问题是, 在空间受限的随机算法中是否需要明确规定运行时间上限。不失一般性, 我们认为空间受限非确定机器的运行步骤最多为其空间复杂度的指数, 因为在由所有可能配置构成的 (有向) 图上, 两种配置间最短路径的上限取决于图的规模 (而这又是空间上限的指数函数)。这种推理对于随机算法并不成立, 因为此时两种配置间的最短路径不再成为配置 1 转化为配置 2 时, 所期望的随机步骤数上限。事实上, 我们很快将看到, 对于对数空间随机算法, 不限制其运行时间上界似乎赋予了这类算法更多的功能, 就是说, 这些 (无限制的) 对数空间随机算法可以 (在指数时间内) 模拟非确定的对数空间运算。模拟过程中重复调用 NL -机, 并在非确定步骤中使用随机选择。如果输入为“是”实例, 则在每一次尝试中, 以至少 2^{-t} 的概率“命中”一个可接受的 t 步 (非确定) 计算, 其中 t 为输入长度的多项式。因此, 随机算法在期望的 2^t 次尝试后接受这个“是”实例。为了拒绝“否”实例 (而不是陷入

死循环),我们希望有一个计数器,在计数达到 2^t (左右)时,如果 2^t 次尝试均失败(没有命中 NL-机的一种可接受计算),则拒绝该输入。这个计数器的实现需要空间 $O(\log t)$ 而非 t (这是容易的)。事实上,有一个“随机计数器”就可以满足需要,如果该计数器能以较高概率计数到大约 2^t 。这样一个计数器的实现留给读者自己完成,见习题 6.18。而利用它可以构造一个随机算法,能(对所有输入)以较高的概率停机,并总是拒绝“否”实例,而以不少于 $1/2$ 的概率接受每个“是”实例。

根据上述讨论,在定义随机的对数空间算法时,明确要求算法在多项式时间内停机。遵从这个习惯, RL 类(或 BPL 类)与 NL 类之间的关系类似于 RP (或 BPP)与 NP 的关系。特别地, RL (或 BPL 类)中概率接受的条件与 RP (或 BPP)相同。

定义 6.11 (RL 和 BPL 类) 称随机的对数空间算法是可接受的,如果该算法总在多项式步后停机。

- 判定问题 S 属于 RL ,如果存在一个可接受的(在线)随机对数空间算法 A ,使得对任意 $x \in S$,有 $\Pr[A(x)=1] \geq 1/2$,而对任意 $x \notin S$,有 $\Pr[A(x)=0]=1$ 。

- 判定问题 S 属于 BPL ,如果存在一个可接受的(在线)随机对数空间算法 A ,使得对任意 $x \in S$,有 $\Pr[A(x)=1] \geq 2/3$,而对任意 $x \notin S$,有 $\Pr[A(x)=0] \geq 2/3$ 。

显然, $RL \subseteq NL \subseteq P$,而 $BPL \subseteq P$ 。注意:即使当允许算法运行于期望的多项式时间并且可以有不停机的运算时, RL 和 BPL 类也保持不变。通过将长的计算截短,并使用(标准的)计数器(可在对数空间内实现),可以很容易地将这种算法转化为可接受的算法。还要注意错误缩减也可在当前环境中使用(并从根本上保持时间和空间上限)。

6.1.5.2 旁观者清

下面给出随机对数空间算法一个有趣的例子。它与判定图的无向连通性有关,我们证明这个问题属于 RL 类(回顾在 5.2.4 节已经证明了这个问题实际上属于 L 类,但算法及分析十分复杂)。作为对比,注意到有向连通性是 NL 完全的(在对数空间归约下)。

为简单起见,考虑如下计算问题:给定无向图 G 及一对结点 (s,t) ,判定在图 G 中 s 和 t 是否连通。注意:(一个给定无向图的)无向连通性的判定可以在对数空间内归约为上述问题(只需检查所有结点对的连通性)。

构造 6.12 对于输入 (G,s,t) ,随机算法从结点 s 开始,经过一个 $\text{poly}(|G|)$ 长的随机漫游,当且仅当经过顶点 t 时接受输入的三元组。这里随机漫游的含义是,在每一步,算法均匀选择当前结点的一个邻居,并移动到该邻居。

观察到这个算法可以在对数空间内实现(因为只需保存当前结点及运行的步数)。显然,如果 s 和 t 在 G 中不连通,则算法总是拒绝 (G,s,t) 。命题 6.13 暗示了图的无向连通性属于 RL 类。

命题 6.13 由图 $G=(V,E)$ 中任一结点出发的长度为 $O(|V| \cdot |E|)$ 的路径,经过所有与起始结点位于同一连通分量的所有结点的概率至少为 $1/2$ 。

从而,这种路径可以在相应的连通分量中漫游(在任何图中)。沿着这条路径,可以到达该连通分量中所有的结点。

证明概要:实际上是要证明,如果 G 是连通的,则开始于结点 s 的随机漫游能以至少 $1/2$ 的概率通过 G 中所有结点。对任意一对结点 (u,v) ,设 $X_{u,v}$ 为一个随机变量,表示开始于结

点 u 的随机漫游在第一次经过 v 时所需步骤数。可以验证, 对任意边 $\{u, v\} \in E$, 有 $E[X_{u,v}] \leq 2|E|$, 见习题 6.19。下面用 $\text{cover}(G)$ 表示开始于 s 的随机漫游通过 V 中所有结点所需步骤数的期望值。我们的目标是求 $\text{cover}(G)$ 的上限。为达到这个目标, 考虑 G 中通过所有结点的任意一条有向回路 C , 注意到

$$\text{cover}(G) \leq \sum_{(u,v) \in C} E[X_{u,v}] \leq |C| \cdot 2|E|$$

特别地, 选择 C 作为 G 的某个生成树上的遍历, 可以得到结论 $\text{cover}(G) < 4 \cdot |V| \cdot |E|$ 。因此, 开始于 s , 长为 $8 \cdot |V| \cdot |E|$ 的随机漫游以至少 $1/2$ 的概率经过 G 中所有结点。□

6.2 计 数

现在讨论一类新的计算问题, 计数问题, 它是对 NP-类判定问题的推广。计数问题是指对可以高效识别的对象数量进行计数。NP 问题的搜索和判定版本为可以高效识别的对象提供了恰当定义, 这又定义了相应的计数问题。

(1) 对任意能高效验证解的搜索问题 (即 $\mathcal{PC}$ 中的二元关系 $R \subseteq \{0,1\}^* \times \{0,1\}^*$ (见定义 2.3)), 考虑对给定实例的解进行计数, 即对输入 x , 要输出 $|\{y: (x,y) \in R\}|$ 。

(2) 对 $\mathcal{NP}$ 中每个判定问题 S 以及每个相应的验证程序 V (见定义 2.5), 考虑对给定实例的 NP-证据进行计数。即对于输入 x , 要输出 $|\{y: V(x,y)=1\}|$ 。

我们考虑这两种类型的计数问题, 以及 (对这些计数问题) 放宽后得到的近似计数问题 (分别见 6.2.1 节和 6.2.2 节)。其他相关问题还包括 “具有唯一解的问题” (见 6.2.3 节) 以及 “均匀生成解” (见 6.2.4 节)。有趣的是, 在有关上述类型计数问题的结论中, 随机程序起着重要作用。

6.2.1 精确计数

继续上述讨论, 定义对可高效识别对象的计数相关的问题类 (回顾 $\mathcal{PC}$ 表示具有多项式长度解且其解可被高效验证的搜索问题, 见定义 2.3)。

定义 6.14 (对可高效识别对象的计数—— $\#P$) $\#P$ 类中包含了可对 $\mathcal{PC}$ 中搜索问题的解计数的所有函数。也就是说, 函数 $f: \{0,1\}^* \rightarrow \mathbb{N}$ 属于 $\#P$, 如果存在 $R \in \mathcal{PC}$, 使得对任意 x , 有 $f(x) = |R(x)|$, 其中 $R(x) = \{y: (x,y) \in R\}$ 。此时, 称 f 是与 R 相关的计数问题, 记作 $\#R$ (即 $\#R = f$)。

$\mathcal{NP}$ 中每个判定问题都可以 Cook-归约到 $\#P$, 这是由于这些问题均可以看作是对某个 $R \in \mathcal{PC}$, 判定 $S_R = \{x: |R(x)| > 0\}$ 的成员 (见 2.1.2 节)。另外, $\mathcal{BPP}$ 类可以 Cook-归约到 $\#P$ (见习题 6.20)。 $\#P$ 类有时也用判定问题定义, 如以下命题中所蕴含的。

命题 6.15 ($\#P$ 的判定版本) 对任意 $f \in \#P$, $S_f \stackrel{\text{def}}{=} \{(x, N): f(x) \geq N\}$ 的成员判定在计算上等价于计算 f 。

实际上, 对任意函数 $f: \{0,1\}^* \rightarrow \mathbb{N}$, 如果存在多项式 p , 使得对任意 $x \in \{0,1\}^*$, 有

$f(x) \leq 2^{p(|x|)}$, 则上述断言对 f 都成立。

证明: 由于关系 R 保证了 $f \in \#P$ (即 $f(x) = |R(x)|$) 为多项式界, 从而存在多项式 p 使得对任意 x 有 $f(x) \leq 2^{p(|x|)}$ 。 S_f 的成员判定易归约为计算 f (即接受输入 (x, N) 当且仅当 $f(x) \geq N$)。利用二分查找法 (见习题 2.9), 可将计算 f 归约为判定 S_f 。这是基于一个事实, 即对于输入 x 及 S_f 的预言访问, 可以通过发出询问 (x, N) 来确定是否有 $f(x) \geq N$ 。注意: 我们已经先验地知道了 $f(x) \in [0, 2^{p(|x|)}]$ 。 ■

计数类 $\#P$ 还涉及到对给定实例的所有可能解进行枚举 (见习题 6.21)。

6.2.1.1 $\#P$ 的功能

前面暗示过, $NP \cup BPP$ 可 (易于) 归约为 $\#P$ 。进一步地, 如定理 6.16 所述, 整个多项式时间层级 (见 3.2 节中的定义) 可以 Cook-归约到 $\#P$ (即 $PH \subseteq P^{\#P}$)。另一方面, $\#P$ 中任意问题都可在多项式空间内解决, 所以 $P^{\#P} \subseteq PSPACE$ 。

定理 6.16 PH 中任意集合都可 Cook-归约到 $\#P$ 。

这里对定理 6.16 不作证明, 因为已知的证明方法都使用了极强的技巧。另外, 证明的主要思想将在定理 6.29 的证明中以更清晰的方式呈现。然而, 附录 F.1 中证明了一个相关结论, 其中暗示 PH 可以通过随机的 Karp-归约归约为 $\#P$ 。

6.2.1.2 $\#P$ 完全性

$\#P$ -完全性的定义类似于 NP -完全性。即一个计数问题 f 是 $\#P$ -完全的, 如果 $f \in \#P$ 且 $\#P$ 中所有问题都可以 Cook-归约为 f 。

我们认为, 与 2.3.3 节中的 NP -完全问题相关的计数问题都是 $\#P$ -完全的。但是这一事实之所以成立并不仅仅出于这些问题的 NP -完全性, 更是由于用于证明其 NP -完全性的归约的一个附加性质。具体来讲, 所使用的 Karp-归约 (或其变形) 具有能保持 NP -证据数量 (以下定义表明了这一点) 的特殊性质。

定义 6.17 (简化的归约) 令 $R, R' \in PC$, g 为从 $S_R = \{x: R(x) \neq \emptyset\}$ 到 $S_{R'} = \{x: R'(x) \neq \emptyset\}$ 的 Karp-归约, 其中 $R(x) = \{y: (x, y) \in R\}$ 而 $R'(x) = \{y: (x, y) \in R'\}$ 。称 g (关于 R 和 R') 是简化的, 如果对任意 x , 有 $|R(x)| = |R'(g(x))|$ 。此时称 g 为 R 到 R' 的简化归约。

我们强调简化的条件涉及两个关系 R 和 R' (而不仅仅是集合 S_R 和 $S_{R'}$)。要求 g 为 Karp-归约似乎有点多余, 因为如果 g 是多项式时间可计算的, 并且对任意 x , 有 $|R(x)| = |R'(g(x))|$, 则这样的 g 本身就构成了由 S_R 到 $S_{R'}$ 的 Karp-归约。特别地, $|R(x)| = |R'(g(x))|$ 蕴含了 $|R(x)| > 0$ (即 $x \in S_R$) 当且仅当 $|R'(g(x))| > 0$ (即 $g(x) \in S_{R'}$)。读者易验证在定理 2.19 证明中使用的 Karp-归约以及 2.3.3 节中的许多归约都是简化的 (见习题 2.29)。

定理 6.18 设 $R \in PC$, 并假设 PC 中任意搜索问题可以简化地归约为 R , 则与 R 相关的计数问题是 $\#P$ -完全的。

证明: 显然与 R 相关的计数问题, 记作 $\#R$, 属于 $\#P$ 。为证明任意 $f' \in \#P$ 可以归约到 f , 考虑关系 $R' \in PC$, 与其相关的计数问题为 f' , 即 $\#R' = f'$ 。由假设可知, 存在 R' 到 R 的简化归约 g 。该归约也将 $\#R'$ 归约为 $\#R$, 即对任意 x , 有 $\#R'(x) = \#R(g(x))$ 。 ■

推论

作为定理 6.18 的直接推论,可以得到,对于给定 CNF 范式的可满足性赋值的计数问题是 $\#P$ -完全的 (因为 R_{SAT} 在简化归约下是 PC -完全的)。类似结论对 2.3.3 节中提到的其他 NP-完全问题也成立,并且事实上对参考文献[85]中列举的所有 NP-完全问题都成立。这些推论来自于这样一个事实,即在 NP-完全问题之间所有的已知归约或者是简化的,或者易于修改为简化的。

我们可以得出结论,与 NP-完全搜索问题相关的许多计数问题都是 $\#P$ -完全的。从而与可高效求解的搜索问题相关的计数问题也可能是 $\#P$ -完全的。

定理 6.19 对于可高效求解的搜索问题,存在与其相关的 $\#P$ -完全计数问题。即存在 $R \in \mathcal{PF}$ (见定义 2.2) 使得 $\#R$ 是 $\#P$ -完全的。

定理 6.19 可以通过人为地构造一个 $\#P$ -完全问题来证明 (见习题 6.22)。以下证明中使用了一个自然的计数问题。

证明: 考虑由对 (Φ, τ) 组成的关系 R_{dnf} , 其中 Φ 为析取范式, τ 为满足 Φ 的真值赋值。注意: 搜索问题 R_{dnf} 是易解的 (如可随机选取一个使输入公式的第一项满足的真值赋值)。为证明 $\#R_{dnf}$ 是 $\#P$ -完全的, 考虑如下由 $\#R_{SAT}$ (由定理 6.18, $\#R_{SAT}$ 是 $\#P$ -完全的) 到 $\#R_{dnf}$ 的归约。给定一个合取范式 Φ , 利用德·摩根律将 $\neg\Phi$ 转化为析取范式 Φ' , 向 $\#R_{dnf}$ 询问 Φ' , 并返回 $2^n - \#R_{dnf}(\Phi')$, 其中 n 表示 Φ (或 Φ') 中的变量个数。定理得证。 ■

思考: 注意定理 6.19 没有使用简化的归约来证明。这并不奇怪, 因为从 $\#R'$ 到 $\#R$ 的简化归约蕴含了 $S_{R'} = \{x: \exists y, \text{使得 } (x, y) \in R'\}$ 可以归约为 $S_R = \{x: \exists y, \text{使得 } (x, y) \in R\}$, 其中 $S_{R'}$ 是 NP-完全的, 而 $S_R \in P$ (由于 $R \in \mathcal{PF}$)。然而, 定理 6.19 的证明与某些判定问题 (即判定一个给定的析取范式是否为恒真命题 (即是否有 $\#R_{dnf}(\Phi') = 2^n$)) 的困难性相关。但是, 是否存在一个 $\#P$ -完全问题是“不基于某些已知的 NP-完全判定问题”呢? 令人惊奇的是, 答案是肯定的。

定理 6.20 对二部图中完美匹配的计数是 $\#P$ -完全的。^①

等价地 (见习题 6.23), 求 0/1 矩阵的积和式 (Permanent) 是 $\#P$ -完全的。一个 $n \times n$ 矩阵 $M = (m_{i,j})$ 的积和式, 记作 $\text{perm}(M)$, 是指对所有 n 元素置换 π , 求乘积 $\prod_{i=1}^n m_{i,\pi(i)}$ 之和。定理 6.20 的证明需要结合以下两个 (多到一) 归约 (分别在命题 6.21 和命题 6.22 中给出), 并利用 $\#R_{3SAT}$ 为 $\#P$ -完全的事实 (见定理 6.18 及习题 2.29)。显然, 以下归约不是简化的。

命题 6.21 3SAT 的计数问题 (即 $\#R_{3SAT}$) 可归约为对整数矩阵求积和式。进一步地, 存在偶数 $c > 0$ 及有限的整数集 I , 使得对于输入的 3CNF 范式 ϕ , 归约生成一个整数矩阵 M_ϕ , 其元素来自于 I , 使得 $\text{perm}(M_\phi) = c^m \cdot \#R_{3SAT}(\phi)$, 其中 m 表示 ϕ 中子句个数。

命题 6.21 的原始证明中使用 $c = 2^{10}$ 和 $I = \{-1, 0, 1, 2, 3\}$ 。可以证明 (见习题 6.24 (其中利用了定理 6.29)), 对任意与 c 互素的整数 $n > 1$, 求模 n 的积和式是 NP-困难的 (在随机归约下)。因此, 这意味着当 $c = 2^{10}$ 时, 对任意奇数 $n > 1$, 求模 n 的积和式是 NP-困难的。相反, 求模 2 的积和式 (相当于计算 mod 2 的行列式) 则是容易的 (即可以在多项式时间内完成, 甚至是在 $\mathcal{NC}$ 内完成)。因此, 假定 $\mathcal{NP} \not\subseteq \mathcal{BPP}$, 则命题 6.21 对奇数 c 不成立 (因为由习

^① 见附录 G.1 中的图论基础。

题 6.24, 求模 2 的积和式是 NP-困难的)。还要注意的, 假设 $P \neq NP$, 则命题 6.21 对于只包含非负整数的 I 可能不成立 (见习题 6.25)。

命题 6.22 求整数矩阵的积和式可归约为求 0/1 矩阵的积和式。进一步地, 该归约将任意整数矩阵 A 映射为 0/1 矩阵 A'' 使得由 A 及 A'' 的积和式, 易求出 A 的积和式。

教学注释: 不建议在课堂上讲述命题 6.21 和命题 6.22 的证明。命题 6.21 证明的高层结构与 NP-问题之间某些精妙的归约风格相似, 但关键是要存在充分的分图。我们还不知道这些分图存在的证据, 也没有任何直觉表明这些分图为何要存在。^①作为替代, 这种分图的存在性可由一个本质上非平凡的特殊构造来证明。因此, 命题 6.21 的证明归结为一个复杂的构造问题, 而其解决方法几乎没有教学上的价值。相反, 命题 6.22 的证明使用了两个简单思想, 在其他环境中也能发挥作用。在适当的提示下, 该证明可以作为一个很好的习题。

命题 6.21 的证明:

使用矩阵 A 的积和式与 A 所代表的加权有向图的圈覆盖 (Cycle Cover) 权重和之间的对应关系。图的圈覆盖是指图中一组无公共顶点的有向圈的简单组合,^②其中包含了图中所有的顶点, 且其权重是相应边权重的乘积。加权有向图的 SWCC 是指其中所有圈覆盖权重之和。

给定一个 3CNF 范式 ϕ , 构造有向加权图 G_ϕ , 使 G_ϕ 的 SWCC 等于 $c^m \cdot \#R_{3SAT}(\phi)$, 其中 c 为全局常量, m 表示 ϕ 中子句个数。不失一般性, 可以假设, ϕ 的每个子句恰含 3 个文字 (不要求这些变量是不同的)。

我们从一个 (关于这种构造的) 高层次描述开始, 其中涉及到 (子句的) 分图, 每个分图包含一些内部顶点及内部 (加权) 边, 此时尚未定义其权值。另外, 每个分图有 3 对指定顶点, 每一对对应着子句中的一个文字, 在每个顶点对中, 一个顶点被指定为进入结点, 另一个为外出结点。图 G_ϕ 中包含 m 个这样的分图, 每个对应着 (ϕ 的) 一个子句, 以及 n 个辅助结点, 对应于 (ϕ 的) 每个变量, 图 G_ϕ 中还包含一些附加的权重为 1 的有向边。具体来讲, 对于每个变量, 引入两个轨迹 (Track), 分别对应于该变量的两种可能的文字。与某个文字相关联的轨迹由一些 (权重为 1 的) 有向边组成, 这些边构成一个简单“圈”, 经过相应的 (辅助) 结点以及对应于该文字出现于各种子句中的指定结点。具体地, 对于该文字的每一次出现, 轨迹由对应于该文字的进入结点进入相应的子句分图, 并由相应的外出结点离开 (如果某个文字在 ϕ 中不出现, 则相应轨迹在对应变量中是一个自环)。如图 6.1 所示, 其中显示了变量 x 的两个轨迹, x 在三个子句中以“正”的形式出现, 在一个子句中以“负”的形式出现。进入结点 (或外出结点) 位于每个分图的上部 (或下部)。

为了让子句分图具有期望的性质, 我们构造一个增强的图, 向每个分图中增加 9 条外部边 (权重为 1), 每条边对应于一对进入和外出结点 (不一定匹配), 该边的方向是从外出结点到进入结点 (图 6.2) (我们强调, 这种辅助构造与前面 G_ϕ 的构造不同, 但在分图的使用上有一定联系)。连接对应于 3 个文字的指定结点对的 3 条边被称为是良好的。称边集 C (如在增强图中由圈构成的集合) 使用了外部边集 S , 如果 C 与 (9 条) 外部边集的交集为 S 。假设以下关于子句分图的 3 条性质成立。

(1) (分图中) 未使用任意外部边 (即使用的外部边集为空集) 的所有圈覆盖的权重和为 0。

① 事实上, 关于这种分图存在性的猜想只能归结为灵感。

② 这里所说的简单圈是指一个强连通的有向图, 其中每个结点有一个进入 (或外出) 边。特别地, 这里也允许自环的存在。

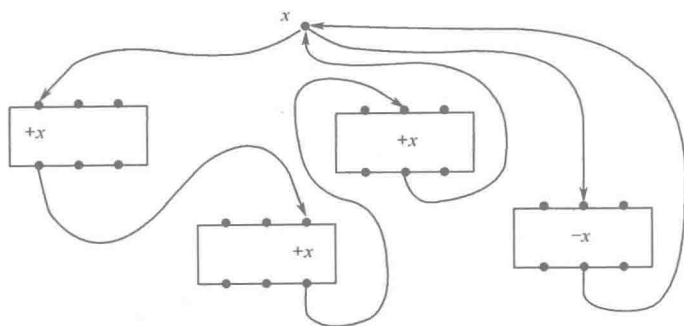


图 6.1 归约为圈覆盖时, 将子句分图相连的轨迹

(2) 令 $V(S)$ 表示依附于 S 的结点集合, 称 S 是良好的, 如果 S 非空, 且 $V(S)$ 中结点可以利用良好的边来完美匹配。^①此时, 存在一个常数 c (实际上就是命题中的 c), 使得对任意良好的集合 S , 所有使用了外部边集 S 的圈覆盖的权重和为 c 。

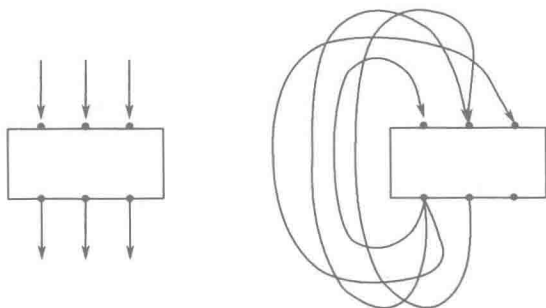


图 6.2 用于分析一个子句分图效果的外部边集。左边是一个附带了轨迹中边的分图 (如构造中所述)。右边是一个分图和分析中使用的 9 条外部边中的 4 条 (其中两条是良好的)

(3) 对于外部边的任意非良好集 $S \neq \emptyset$, 所有使用了外部边集 S 的圈覆盖权重之和为 0。

注意: 上述 3 项包括了所有可能的情况。还要注意的, 由一个圈覆盖 (增强后的分图) 使用的外部边集必为一个匹配 (即这些边一定无公共结点)。

直观上, 在 (增强后的分图) 外部边的良好集与经过 (未增强) 分图的轨迹中的边对之间存在对应关系。事实上在 G_ϕ 中使用了未增强的分图。利用上述 (未增强) 分图的性质, 可以证明 ϕ 的每种可满足赋值对于 G_ϕ 的 SWCC 的贡献恰为 c^m (见习题 6.26)。从而, G_ϕ 的 SWCC 等于 $c^m \cdot \#R_{3SAT}(\phi)$ 。

在证明了抽象归约的正确性之后, 再来讨论子句分图的实现。第一种实现是 Deus ex 机, 图 6.3 描绘了其邻接矩阵。其正确性 (当 $c=12$ 时) 可以通过计算相应子矩阵的积和式来验证 (见习题 6.28 中的类似分析)。

图 6.4 描绘了子句分图的一种结构性更强的实现, 其中涉及到后面将要实现的 (六边形) 盒子。盒中包括几个结点以及加权边, 但是只有两个称为终端的结点与外部相连 (图 6.4)。

^① 显然, 任意由良好边构成的非空集合都是一个良好集合。因此, 单个集合是良好的当且仅当相应边是良好的。另一方面, 任意由 3 条 (结点不相交的) 外部边构成的集合 S 也是良好的, 因为利用所有的 3 条良好边, 可构造 $V(S)$ 的完美匹配。因此, 良好集的概念只对两条边构成的集合是“非平凡的”。这样的集合 S 是良好的当且仅当 $V(S)$ 中包含两对相应的指定结点。

分图中有8个结点,前6个为指定(进入和外出)结点。与第*i*个文字相关联的进入结点(或外出结点)标记为*i*(或*i+3*)。相应的邻接矩阵如下。

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 2 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

注意:第3条~第6条边可以压缩,但是此时得到的含7个结点的图将(不一定在严格意义上)不符合分图的定义,其中要求6个指定结点应该是不同的。

图 6.3 向圈覆盖的归约中使用的 Deus ex 机的子句分图

子句分图由该盒子的5个副本构成,其中3个副本被指定为子句中的3个文字(标记为LB1、LB2、LB3),图6.4中还画出了附加的结点和边。特别地,子句分图包含6个上述的指定结点(即每个文字的一对进入和外出结点)、2个附加结点(在图的左右两端)以及一些边(权重均为1)。每个指定结点都有一个自环,并依附于一个附加边,当该结点为进入(或外出)结点时,这条边为外出(或进入)边。每个盒子中,与某些文字相关的两个端点与相应的指定结点相连(如entry1的外出边与LB1盒的右端点相连)。注意:5个盒子位于同一条路径上(由左向右),而所有反向边都位于该路径下方。

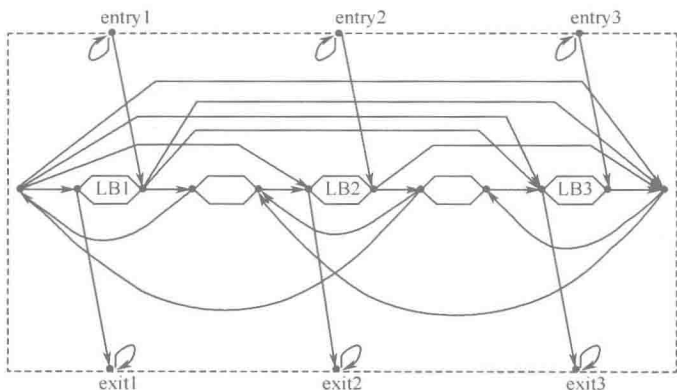


图 6.4 向圈覆盖归约时使用的结构化子句分图

继续上述讨论,需要说明盒子应该具备何种性质。这里需再次考虑增强型盒子,其中向特定结点加入外部边(权重为1)。此时(图6.5),有一对将盒子两端相连的非平行边,以及两个自环(位于每个端点)。假设盒子具有以下3个性质。

- (1) 没有使用任意外部边的(盒子的)所有圈覆盖权重之和为0。
- (2) 存在一个常数*b*(在上述例子中,*b*=4)使得对两个非平行边中的任意一条,使用该边的圈覆盖权重之和为*b*。
- (3) 对任意(非空)的自环集合*S*,使用*S*的(盒子的)所有圈覆盖权重之和为0。

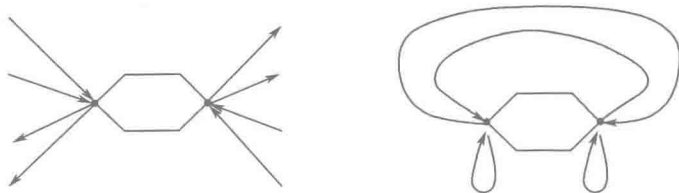


图 6.5 用于分析盒子效果的外部边。左边为一个有邻接边的盒子（如分图构造中）。右边为一个盒子及四条用于分析的外部边

注意：上述 3 项包括了所有可能的情况。可以证明，关于盒子的条件暗示了图 6.4 中的构造满足关于子句分图的假设（见习题 6.27）。具体地，此时，有 $c = b^5$ 。关于盒子自身，以下的 4×4 的邻接矩阵给出了一个较小的 Deus ex 机。

$$\begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 3 & 0 \end{pmatrix} \quad (6.4)$$

其中两个端点对应于第一个和第四个结点。其正确性（ $b=4$ 时）可通过计算相应子矩阵的积和式来验证（见习题 6.28）。 ■

命题 6.22 的证明：证明包括两步。第一步，证明求整数矩阵的积和式可归约为计算非负矩阵的积和式。归约过程如下。对于一个 $n \times n$ 的整数矩阵 $A = (a_{i,j})_{i,j \in [n]}$ ，令 $\|A\|_\infty = \max_{i,j} (|a_{i,j}|)$ 以及 $Q_A = 2(n!) \cdot \|A\|_\infty + 1$ 。注意：对于给定 A ， Q_A 的值可在多项式时间内计算，特别地， $\log_2 Q_A < n^2 \log \|A\|_\infty$ 。给定 A ，归约中构造非负矩阵 $A' = (a_{i,j} \bmod Q_A)_{i,j \in [n]}$ （即 A' 中元素来自 $\{0, 1, \dots, Q_A - 1\}$ ），向预言机询问 A' 的积和式，并当 $v < Q_A/2$ 时输出 $v = \text{perm}(A') \bmod Q_A$ ，否则，输出 $-(Q_A - v)$ 。一个重要的观察是 $\text{perm}(A) \equiv \text{perm}(A') \pmod{Q_A}$ ，而 $|\text{perm}(A)| \leq (n!) \cdot \|A\|_\infty^n < Q_A/2$ 。

因此， $\text{perm}(A') \bmod Q_A$ （属于 $\{0, 1, \dots, Q_A - 1\}$ ）确定了 $\text{perm}(A)$ 。注意： $\text{perm}(A')$ 可能远远大于 $Q_A > |\text{perm}(A)|$ ，不过 $\text{perm}(A')$ 与 $\text{perm}(A)$ 模 Q_A 是相等的。

第二步，证明非负矩阵积和式的计算可归约为计算 0/1-矩阵的积和式。在这个归约中，将计算积和式看作是求相应加权有向图中所有圈覆盖的权重（SWCC）之和（见命题 6.21 的证明）。因此，将计算具有非负权值的有向图的 SWCC 归约为计算无权、无平行边的有向图的 SWCC（这对应于 0/1-矩阵）。归约中进行保持 SWCC 值不变的本地代替。这些本地代替组合了以下两个本地代替（保持 SWCC 不变）。

(1) 将权重为 $w = \prod_{i=1}^t w_i$ 的边用一条长为 t 的路径（即 $t-1$ 个内部结点）和所有内部结点的自环（权重为 1）代替，路径中边的权重分别为 $w_1, w_2, \dots, w_t$ 。

注意：一个使用了原始边的圈覆盖对应着使用整条路径的圈覆盖，而一个不使用原始边的圈覆盖对应于使用所有自环的圈覆盖。

(2) 将权重为 $w = \prod_{i=1}^t w_i$ 的边用 t 条平行的长为 2 的路径代替，且满足第 i 条路径中的第一条边权重为 w_i ，第二条边权重为 1，而中间结点有一个自环（权重为 1）（这里使用长为 2 的路径是因为不允许有平行边）。

注意：使用原始边的圈覆盖对应于一组使用 t 条路径之一的圈覆盖（以及所有中间结点的自环），而一个不使用原始边的圈覆盖对应于使用所有自环的圈覆盖。

特别地，可将每个正的权重 w ，二进制表示为 $\sigma_{|w|-1} \cdots \sigma_0$ ，写作 $\sum_{i:\sigma_i=1} (1+1)^i$ ，并利用充分的代替（即首先对外部和（对所有 $\{i:\sigma_i=1\}$ ）使用加法代替，然后对每个幂 2^i 使用乘法代替，最后对每个 $1+1$ 使用加法代替）。将这个过程应用于第一步中得到的矩阵 A' ，我们可以高效地得到一个 0/1-矩阵 A'' ，使得 $\text{perm}(A') = \text{perm}(A'')$ （特别地， A'' 的维数是 A' 二元代替长度的多项式，后者又是 A 的二元代替长度的多项式）结合两个归约（步骤），命题得证。■

6.2.2 近似计数

我们已经看到，（对 PC 中关系）进行精确计数似乎比解相应的搜索问题更难。本小节放宽对计数问题的要求。首先简单介绍具有（较大）加法偏移的近似，然后讨论相关近似。

考虑与任意一个关系 $R \in PC$ 相关的计数问题。不失一般性，假设该问题所有 n 比特实例的解长度均相同，为 $l(n)$ ，其中 l 为多项式。要注意，计算 $\#R$ 有可能是困难的，但对给定 x ，当允许最大加法误差为 $0.01 \cdot 2^{l(|x|)}$ 时，近似地求 $\#R(x)$ 是容易的（通过在 x 的解中随机采样）。事实上，这样的近似十分粗糙，但它不是平凡的（因为我们还不知道该如何通过确定方法得到这种近似条件）。大体上讲，可以高效地随机产生 $\#R(x)$ 的近似值，它以较高概率与精确值相差最多一个相加项，这个差值与解的绝对上界（即 $2^{l(|x|)}$ ）有关。

命题 6.23（有较大加法偏移的近似） 设 $R \in PC$ ， l 为多项式，且满足 $R \subseteq \bigcup_{n \in \mathbb{N}} \{0,1\}^n \times \{0,1\}^{l(n)}$ ，则对任意多项式 p ，存在一个概率多项式时间算法 A ，使得对任意 $x \in \{0,1\}^*$ 及 $\delta \in \{0,1\}$ ，有

$$\Pr \left[|A(x, \delta) - \#R(x)| > \left(1/p(|x|) \right) \cdot 2^{l(|x|)} \right] < \delta \quad (6.5)$$

通常 δ 为二元表示，从而 $A(x, \delta)$ 的运行时间上界为 $\text{poly}(|x| \cdot \log(1/\delta))$ 。

证明概要：对于输入的 x 和 δ ，算法 A 设置 $t = \Theta \left(p(|x|)^2 \cdot \log(1/\delta) \right)$ ，均匀选择 $y_1, y_2, \dots, y_t$ 并输出 $2^{l(|x|)} \cdot \left| \{i : (x, y_i) \in R\} \right| / t$ 。□

讨论：当 $\#R(x) > (1/p(|x|)) \cdot 2^{l(|x|)}$ 时，命题 6.23 对某些 x 成立，否则，将得到符合式 (6.5) 中上限的一种平凡近似（即输出常数 0）。与这种加法近似相反，相关因子近似更有意义。具体而言，我们感兴趣的是对 $\#R(x)$ 乘以一个常数因子（或其他某些合理的因子）的近似。在 6.2.2.1 节中讨论了一种自然的 $\#P$ -完全问题，可在概率多项式时间内得到该问题的相关近似。不能期望 $\#P$ 中所有计数问题都有这种现象，因为相关近似考虑到了区分有解实例与无解实例（即判定 S_R 成员可归约为 $\#R$ 的相关近似）。因此， $\#P$ 中所有问题的相关近似至少与 $\mathcal{NP}$ 中所有判定问题一样难。然而，在 6.2.2.2 节中，我们将证明前者并不比后者更难，即 $\#P$ 中所有问题的相关近似可通过到 $\mathcal{NP}$ 的随机 Cook 归约而得到。在介绍这些结论之前，给出如下定义（实际上可以将其强化为在近似因子 $1+\varepsilon$ ， $\varepsilon \in (0,1)$ ，之内）^①。

① 这里对任意的 F ，没有给出 F -因子近似的形式化定义，虽然在几次非正式的讨论中用到了这个概念。这个术语有几种定义（在这些非正式讨论中这些定义是等价的）。例如， $\#R$ 的一个 F -因子近似可能是指，输出的 $A(x)$ 以很高的概率满足 $\#R(x)/F(|x|) \leq A(x) \leq F(|x|) \cdot \#R(x)$ 。另外，还可以要求 $\#R(x) \leq A(x) \leq F(|x|) \cdot \#R(x)$ （或者要求 $\#R(x)/F(|x|) \leq A(x) \leq \#R(x)$ ）。

定义 6.24 (带相关偏移的近似) 设 $f: \{0,1\}^* \rightarrow \mathbb{N}$, $\varepsilon, \delta: \mathbb{N} \rightarrow [0,1]$, 随机过程 Π 称为 f 的 (ε, δ) -近似, 如果对任意 x , 有

$$\Pr \left[|\Pi(x) - f(x)| > \varepsilon(|x|) \cdot f(x) \right] < \delta(|x|) \quad (6.6)$$

称 f 可被高效地 $(1-\varepsilon)$ 近似 (或 $(1-\varepsilon)$ 近似), 如果存在概率多项式算法 A , 能构成 f 的 $(\varepsilon, 1/3)$ -近似。

算法 A (错误概率为 $1/3$) 的错误概率可在 $O(\log(1/\delta))$ 次迭代后缩小为 δ (见习题 6.29)。

典型地, A 的运行时间为 $1/\varepsilon$ 的多项式, ε 称为偏移参数。

我们指出定义 6.24 中蕴含的计算问题是求函数 f 的 $(1-\varepsilon)$ 近似 (即解搜索问题 $R_{f,\varepsilon} \stackrel{\text{def}}{=} \{(x,v): |v - f(x)| \leq \varepsilon(|x|) \cdot f(x)\}$)。典型地 (细节见习题 6.30), 该搜索问题在计算上等价于区分集合 $\{(x,v): v < (1-\varepsilon(|x|)) \cdot f(x)\}$ 与集合 $\{(x,v): v > (1+\varepsilon(|x|)) \cdot f(x)\}$ 中成员的承诺 (“缝隙”) 问题。

6.2.2.1 $\#R_{\text{dnf}}$ 的相关近似

本节提出一个自然的 $\#P$ -完全问题, 可在概率多项式时间内找到其常数因子近似。关于非自然 $\#P$ -完全问题的更强的结论可见习题 6.31。

考虑由对 (Φ, τ) 组成的关系 R_{dnf} , 其中 Φ 为析取范式, τ 为满足 Φ 的一种真值赋值。回顾前面讲过 R_{dnf} 的搜索问题是易解的, 而定理 6.19 的证明中指出 $\#R_{\text{dnf}}$ 是 $\#P$ -完全问题 (通过一个非简化的归约)。然而, 我们仍将看到, 存在一个概率多项式时间算法, 能求出 $\#R_{\text{dnf}}$ 的常数因子近似。要注意的一个事实是, $\#R_{\text{dnf}}$ 在非简化归约下是 $\#P$ -完全的, 这意味着 $\#R_{\text{dnf}}$ 的常数因子近似并不表明 $\#P$ 中所有问题都有类似的近似。事实上, 我们不能期望 $\#P$ 中每个问题都有 (概率) 多项式时间常数因子近似算法, 因为这将蕴含着 $\mathcal{NP} \subseteq \mathcal{BPP}$ (因为常数因子近似允许区分有解的实例与无解的实例)。

$\#R_{\text{dnf}}$ 的近似算法可以通过将其 $(\varepsilon, 1/3)$ -近似确定地归约到命题 6.23 中的 (加法偏移) 近似而得到。考虑析取范式 $\phi = \bigvee_{i=1}^m C_i$, 其中每个 $C_i: \{0,1\}^n \rightarrow \{0,1\}$ 是一个合取式。我们的任务是对至少使一个合取式满足的赋值数量求近似值。实际上, 要解决的是一个更普遍的问题, 那就是 (非显式地) 给定 m 个子集 $S_1, S_2, \dots, S_m \subseteq \{0,1\}^n$, 求 $|\bigcup_i S_i|$ 的近似值。此时, S_i 表示满足合取式 C_i 的真值赋值集合。通常对这些集合有两个计算上的假设 (其中 “高效” 的意思是可在关于 $n \cdot m$ 的多项式时间内实现)。

(1) 给定 $i \in [m]$, 可以高效地确定 $|S_i|$ 。

(2) 给定 $i \in [m]$ 和 $J \subseteq [m]$, 可以高效地求出加法偏移在 $1/\text{poly}(n+m)$ 以内的 $\Pr_{s \in S_i} \left[s \in \bigcup_{j \in J} S_j \right]$ 的近似值。

计算环境满足这些假设 (其中 $S_i = C_i^{-1}(1)$, 见习题 6.32)。求 $|\bigcup_{i=1}^m S_i|$ 近似值的关键之处在于

$$\left| \bigcup_{i=1}^m S_i \right| = \sum_{i=1}^m \left| S_i \setminus \bigcup_{j < i} S_j \right| = \sum_{i=1}^m \Pr_{s \in S_i} \left[s \notin \bigcup_{j < i} S_j \right] \cdot |S_i| \quad (6.7)$$

对式 (6.7) 中的概率可以通过假设 (2) 来求近似值。这样就得到了如下算法, 其中 ε 表示期

望的偏移参数 (即希望得到 $(1 \pm \varepsilon) \cdot \left| \bigcup_{i=1}^m S_i \right|$)。

构造 6.25: 令 $\varepsilon' = \varepsilon / m$, 对 $i=1$ 到 m , 做如下计算。

(1) 利用假设 1, 求 $|S_i|$ 。

(2) 利用假设 2, 得到近似 $\tilde{p}_i = p_i \pm \varepsilon'$, 其中 $p_i \stackrel{\text{def}}{=} \Pr_{s \in S_i} \left[s \notin \bigcup_{j < i} S_j \right]$ 。令 $a_i \stackrel{\text{def}}{=} \tilde{p}_i \cdot |S_i|$,

输出所有 a_i 之和。

令 $N_i = p_i \cdot |S_i|$, 并注意到由式 (6.7), 有 $\left| \bigcup_i S_i \right| = \sum_i N_i$ 。我们感兴趣的是 $\sum_i a_i$ 提供的对于 $\sum_i N_i$ 的近似程度。利用 $a_i = (p_i \pm \varepsilon') \cdot |S_i| = N_i \pm \varepsilon' \cdot |S_i|$ (对每个 i), 可以得到 $\sum_i a_i = \sum_i N_i \pm \varepsilon' \cdot \sum_i |S_i|$ 。利用 $\sum_i |S_i| \leq m \cdot \left| \bigcup_i S_i \right| = m \cdot \sum_i N_i$ (以及 $\varepsilon = m\varepsilon'$), 可以得到 $\sum_i a_i = (1 \pm \varepsilon) \sum_i N_i$ 。因此, 有如下结论 (见习题 6.32)。

命题 6.26 对任意正多项式 p , R_{dnf} 的计数问题可以高效地求 $(1 - (1/p))$ -近似。

利用定理 6.19 证明中的归约, 得到结论, 对于给定合取范式的不可满足的真值赋值的计数问题可以高效地求 $(1 - (1/p))$ -近似。然而, 要注意, 对于这种范式可满足的真值赋值的计数问题则不能被高效地近似。同时, 还有一个普遍现象, 即相关近似在一种量级中是可能的, 但在与其互补的量级中则不可能。当然, 这种现象不会出现在加法偏移近似中。

6.2.2.2 #P 类的相关近似

前面提到, 我们无法期望能高效地近似计算每个 #P 问题, 而在本节其余的部分, “近似”一词是 “相关近似” (见定义 6.24) 的简称。特别地, 高效的近似 #R 可以得到判定 $S_R = \{x : R(x) \neq \Phi\}$ 中成员的高效算法。因此, 最多只能希望求 #R 的近似不比判定 S_R 更难 (即求 #R 的近似可在多项式时间内归约到 S_R)。实际上, 每个 NP-完全问题 (即如果 S_R 是 NP-完全的) 都是这样的。更一般地, 我们将证明, #P 中任意问题的近似可以在概率多项式时间内归约为 NP。

定理 6.27 对任意 $R \in PC$ 以及正多项式 p , 存在一个概率多项式时间预言机, 在给定到 NP 的预言访问时, 能构成 #R 的 $(1/p, \mu)$ -近似, 其中 μ 为可忽略的函数 (如 $\mu(n) = 2^{-n}$)。

对任意常数 $\delta < 0.5$, 由于可使用错误归约 (见习题 6.29), 所以注意只要构成 #R 的 $(1/p, \delta)$ -近似即可。同时, 也可证明只要构成 #P 中每个问题的 $(1/2, \delta)$ -近似即可 (见习题 6.33)。

教学注释: 以下证明要利用附录 D.2 中 Hash 函数的概念。具体而言, 我们假设读者熟悉其基本定义 (附录 D.2.1), 至少一种构造 (附录 D.2.2) 以及引理 D.4 (附录 D.2.3)。附录 D.2.3 中更前沿的内容 (引理 D.4 之后) 在本节不会涉及 (但是 6.2.4.2 节会用到其中一部分)。

证明: 给定 x , 证明如何求 $|R(x)|$ 在某个常数因子范围内的近似。期望的 $(1 - (1/p))$ -近似可以利用习题 6.33 中的方法求得。我们还假设 $R(x) \neq \Phi$, 从询问 “ x 是否在 S_R 中” 开始, 如果答案为否定则停机 (并输出 0)。不失一般性, 假设 $R(x) \subseteq \{0, 1\}^l$, 其中 $l = \text{poly}(|x|)$, 主要求某个 $i \in \{1, 2, \dots, l\}$ 使得 $2^{i-4} \leq |R(x)| \leq 2^{i+4}$ 。

计算过程以迭代方式进行。对 $i=1, 2, \dots, l+1$, 检查是否有 $|R(x)| < 2^i$ 。如果回答是肯定的,

则停机并输出 2^i ，否则，继续下一次迭代（事实上，如果能得到对所有这些询问的正确回答，则输出的 2^i 将满足 $2^{i-1} \leq |R(x)| < 2^i$ ）。

当然，这里的关键问题是如何检查 $|R(x)| < 2^i$ 是否成立。方法是对集合 $R(x)$ 使用一个“随机筛”，使得每个元素以 2^{-i} 的概率通过筛，从而，我们期望 $R(x)$ 中有 $|R(x)|/2^i$ 个元素能通过筛。假设 $R(x)$ 中通过随机筛的元素个数确实为 $\lfloor |R(x)|/2^i \rfloor$ ，则当且仅当 $R(x)$ 中某些元素通过筛时有 $|R(x)| \geq 2^i$ 。假设这个筛可以高效地实现，则 $R(x)$ 中是否有元素通过筛是一个“NP-型”问题（因此，可以在 NP-预言中使用）。结合两个假设，可以使得只要 $R(x)$ 中有元素通过筛，则进行下一次迭代。进一步地，即使 $R(x)$ 中通过随机筛的元素个数约为 $|R(x)|/2^i$ ，这种实现也可以提供一个合理的近似。事实上，这种方法提供的近似水平与随机筛的近似程度密切相关。具体细节如下。

随机筛的实现

随机筛可以用一类 Hash 函数来实现（见附录 D.2）。具体地，在第 i 次迭代中，使用函数类 H_l^i ，使得每个 $h \in H_l^i$ 具有 $\text{poly}(l)$ -比特的描述长度，并可将 l 比特长的串映射为 i 比特长的串。进一步地，该类与一个高效的求值算法相伴（即将对 (h, x) 映射到 $h(x)$ ）并（对每个 $S \subseteq \{0,1\}^l$ ）满足

$$\Pr_{h \in H_l^i} \left[\left| \{y \in S : h(y) = 0^i\} \right| \notin (1-\varepsilon, 1+\varepsilon) \cdot 2^{-i} |S| \right] < \frac{2^i}{\varepsilon^2 |S|} \quad (6.8)$$

（见引理 D.4）。随机筛令 y 通过当且仅当 $h(y) = 0^i$ 。事实上，这个随机筛并没有上述讨论中假设的那样完美，但式 (6.8) 表明，在某种意义上，它已经足够好了。特别地，式 (6.8) 蕴含了如果 $i \leq \log_2 |S| - O(1)$ ，则 S 中某个串很有可能通过筛，而当 $i \geq \log_2 |S| + O(1)$ 时， S 中不存在能通过筛的串。

询问的实现

回顾对某些 x 、 i 和 $h \in H_l^i$ ，我们需要确定是否有 $\{y \in R(x) : h(y) = 0^i\} = \emptyset$ 。这种类型的问题可以理解为是对集合

$$S_{R,H} \stackrel{\text{def}}{=} \{(x, i, h) : \exists y, \text{ 使得 } (x, y) \in R \wedge h(y) = 0^i\} \quad (6.9)$$

的成员判定。

利用 $R \in PC$ 的假设以及 Hash 函数类有高效的求值算法这一事实，可以证明 $S_{R,H}$ 属于 NP 。

实际的程序

对于输入 $x \in S_R$ 以及 $S_{R,H}$ 的预言访问，程序以迭代方式运行，从 $i=1$ 开始，在 $i=l$ 时停止（如果之前没有停止），其中 l 表示 x 的可能解的长度。在第 i 次迭代中（ $i < l$ ），随机均匀选择 $h \in H_l^i$ 并发出预言询问：是否 $(x, i, h) \in S_{R,H}$ 。如果应答为否定，则停止并输出 2^i ，否则，继续下一次迭代（ $i \leftarrow i+1$ ）。当然，在到达最后一次迭代时（即 $i=l$ ），停机并输出 2^l 。

实际上，我们忽略了 $x \notin S_R$ 的情况，这可通过对上述过程稍做修改来处理。具体地，对于输入 x ，首先向 S_R 询问 x ，并当应答为否定时输出 0，否则，继续上述过程。

分析

分别求程序输出值太小的概率上限和输出值太大的概率上限。根据上述讨论,可以假设 $|R(x)| > 0$, 并令 $i_x = \lfloor \log_2 |R(x)| \rfloor \geq 0$ 。直观上, 在第 i ($i < i_x$) 次迭代中, 期望 $R(x)$ 中(至少)有 2^{i_x-i} 个元素通过筛, 从而在进行第 $i_x - O(1)$ 次迭代之前, 不可能停机。类似地, 也不可能到达第 $i_x + O(1)$ 次迭代, 因为此时期望 $R(x)$ 中没有元素能通过筛(实际上的期望值为 $2^{-O(1)}$)。以下(针对两种情况)给出更严格的分析。

(1) 程序在第 i 次迭代 ($i < i_x$) 停机的概率为 $\Pr_{h \in H_i^t} \left[\left| \{y \in R(x) : h(y) = 0^i\} \right| = 0 \right]$, 其上限为 $2^i / |R(x)|$ (由式 (6.8) 可知, 并令 $\varepsilon = 1$)^①。因此, 程序在第 $i_x - 3$ 次迭代之前停机的概率上限是 $\sum_{i=0}^{i_x-4} 2^i / |R(x)|$, 小于 $1/8$ (因为 $i_x \leq \log_2 |R(x)|$), 从而, 输出不小于 $2^{i_x-3} > |R(x)|/16$ 的概率至少为 $7/8$ (因为 $i_x > \log_2 |R(x)| - 1$)。

(2) 程序在第 i 次迭代 ($i > i_x$) 中不停机的概率为 $\Pr_{h \in H_i^t} \left[\left| \{y \in R(x) : h(y) = 0^i\} \right| \geq 1 \right]$, 其上限为 $\alpha / (\alpha - 1)^2$, 其中 $\alpha = 2^i / |R(x)| > 1$ (由式 (6.8) 可知, 并令 $\varepsilon = \alpha - 1$)^②。因此, 程序在第 $i_x + 4$ 次迭代中不停机的概率上限为 $8/49 < 1/6$ (因为 $i_x > \lfloor \log_2 |R(x)| \rfloor - 1$), 从而, 输出不超过 $2^{i_x+4} \leq 16 \cdot |R(x)|$ 的概率至少是 $5/6$ (由于 $i_x \leq \log_2 |R(x)|$)。

因此, 上述程序输出一个值 v 满足 $v/16 \leq |R(x)| < 16v$ 的概率至少为 $(7/8) - (1/6) > 2/3$ 。利用习题 6.33 中的思想可以减少偏差(并减少错误概率, 如习题 6.29), 定理得证。 ■

思考

定理 6.27 证明中一个重要现象是, 我们无法直接检查解的数量是否大于某个给定整数(即使有 NP-预言的帮助), 只能检查(在有 NP-预言帮助时)“通过一个随机筛”的解的数量是否大于 0。由于通过随机筛的解的数量反映了全部解的数量(以筛的密度为标准), 由此得到求解的数量近似值的一种方法。

我们指出, 也可以检查“通过随机筛”的解的数量是否大于一个小整数, 这里“小”的含义是输入长度的多项式(见习题 6.35)。具体地, 检查的复杂度是阈值大小的线性函数, 而不是其二进制表示长度的线性函数。事实上, 在许多情况下, 使用的阈值大小为某个效率参数的多项式(而不是令阈值为 0)更具优势。这方面的例子可见于 6.2.4.2 节及参考文献[106]中。

6.2.3 对唯一解的搜索

在讨论搜索问题解的数量时, 一个自然的计算问题是对只有一个解的实例与无解的实例进行区分(或者当这个唯一解存在时找到唯一解)。我们指出, 具有唯一解的实例使大量的讨论更加方便(如习题 6.24 和 10.2.2.1 节)。严格意义上, 搜索或判定唯一解的存在性是用承诺

① 注意: 0 不在开区间 $(0, 2\rho)$ 中, 其中 $\rho = |R(x)|/2^i > 0$ 。

② 这里使用了一个事实, 即 $1 \notin (2\alpha^{-1} - 1, 1)$, 利用如下假设可以得到更好的界限, 即对任意 y , 当 h 在 H_i^t 中均匀选择时, $h(y)$ 的值在 $\{0, 1\}^i$ 中均匀分布。在这种情况下, $\Pr_{h \in H_i^t} \left[\left| \{y \in R(x) : h(y) = 0^i\} \right| \geq 1 \right]$ 的上界为 $E_{h \in H_i^t} \left[\left| \{y \in R(x) : h(y) = 0^i\} \right| \right] = |R(x)|/2^i$ 。

问题（见 2.4.1 节）框架来定义的。

定义 6.28（具有唯一解实例的搜索和判定问题） 二元关系 R 具有唯一解的实例集合定义为 $US_R \stackrel{\text{def}}{=} \{x: |R(x)|=1\}$ ，其中 $R(x) \stackrel{\text{def}}{=} \{y: (x,y) \in R\}$ 。按照惯例，定义集合 $S_R = \{x: |R(x)| \geq 1\}$ 以及 $\bar{S}_R \stackrel{\text{def}}{=} \{0,1\}^* \setminus S_R = \{x: |R(x)|=0\}$ 。

• 求 R 的唯一解定义为带承诺 $US_R \cup \bar{S}_R$ 的搜索问题 R （见定义 2.29）。

续定义 2.30， R 的唯一解的公正搜索定义为带承诺 US_R 的搜索问题 R 。

• 判定 R 的唯一解的定义为承诺问题 $(US_R, \bar{S}_R)$ （见定义 2.31）。

有趣的是，在许多情况下，承诺不会使这些问题比无承诺的原始问题更容易。也就是说，对所有已知的 NP-完全问题，原始问题可以在概率多项式时间内归约为相应的唯一实例问题。

定理 6.29 设 $R \in PC$ ，并假设 PC 中每个搜索问题都可以简化归约到 R ，则求解 R 的搜索问题（或 S_R 的成员判定）可在概率多项式时间内归约为求 R 的唯一解（或承诺问题 $(US_R, \bar{S}_R)$ ）。进一步地，存在一个概率多项式时间内计算的映射 M ，使得对任意 $x \in \bar{S}_R$ ，有 $\Pr[M(x) \in \bar{S}_R] = 1$ ，而对任意 $x \in S_R$ ，有 $\Pr[M(x) \in US_R] \geq 1/\text{poly}(|x|)$ 。

我们强调一个事实，上述假设实际上称 R 在简化的归约下是 PC -完全的，这对于定理 6.29 至关重要（见习题 6.36）。随机 Karp-归约 M 较大的错误概率（但不为 1）可以通过迭代而减小，从而得到一个随机 Cook-归约，具有指数衰减的错误概率。注意：最终得到的归约可能会发出许多违背承诺的询问，并仍可根据符合承诺的询问（以高概率）做出正确回答（具体地，在搜索问题中，将通过检查得到的解而摒弃错误解，而在判定问题中，依赖于一个事实，即对任意 $x \in \bar{S}_R$ ，恒有 $M(x) \in \bar{S}_R$ ）。

证明：主要证明定理的后一部分（从而其主要部分可直接推论）。证明中用到定理 6.27 的证明思想，仍大量使用 Hash 函数，并引用附录 D.2.1-D.2.2 中的内容。

与定理 6.27 的证明相同，主要思想是对 $R(x)$ 使用一个“随机筛”，此时希望只有一个元素通过筛。具体地，如果令每个元素通过筛的概率约为 $1/|R(x)|$ ，则单个元素通过的概率为常数。此时，可得到具有唯一解的实例（即具有唯一 NP-证据的 $S_{R,H}$ 的实例），这样就（基本）满足了定理的要求。随机筛利用一个随机的函数实现，该函数选自于一个足够大的 Hash 函数类（见附录 D.2）。此时出现以下几个问题：

(1) 如何估算 $|R(x)|$ ？注意：我们需要利用 $|R(x)|$ 的近似值来确定足够大的 Hash 函数类。此时，调用定理 6.27 也无法得到这个近似值，因为其中的预言机使用一个 NP 预言（这种预言很难获取，更不用说归约中还会发出多个预言询问）。^①因此，作为代替方案，可以均匀选择 $m \in \{0,1,\dots,\text{poly}(|x|)\}$ ，并注意（当 $|R(x)| > 0$ 时）令 $\Pr[m = \lceil \log_2 |R(x)| \rceil] = 1/\text{poly}(|x|)$ 。然后，随机地将 x 映射为 (x,m,h) ，其中 h 从一个充分的 Hash 函数类中均匀选取。

(2) 如何将 $R(x)$ 中单个元素是否通过随机筛的问题转化为 R 的唯一解实例？回顾在定理 6.27 的证明中， $R(x)$ 中通过筛的元素集合（记作 h ）的非空性质可以通过检查 $S_{R,H} \in NP$ （由式 (6.9) 定义）的成员 (x,m,h) 来确定。或者说， $(x,m,h) \in S_{R,H}$ 的 NP-证据数量等于

^① 当然，两个问题都可利用到具有唯一解实例的归约来求解，但是我们尚未得到这样一种规约——下面就来构造它。

$R(x)$ 中通过筛的元素个数。因此, $R(x)$ 中单个元素通过筛(记作 h)当且仅当 $(x, m, h) \in S_{R,H}$ 具有唯一的 NP-证据。利用 $S_{R,H}$ 到 S_R 的简化归约(由定理中的假设保证), 可以得到所需实例。

注意: 当 $R(x) = \emptyset$ 时, 上述映射总是生成一个“否”实例($S_{R,H}$ 的实例, 从而也是 S_R 的实例)。具体细节如下。

实现(即映射 M)

与定理 6.27 的证明相同, 不失一般性, 假设 $R(x) \subseteq \{0,1\}^l$, 其中 $l = \text{poly}(|x|)$ 。首先均匀地选择 $m \in \{1, 2, \dots, l+1\}$ 及 $h \in H_l^m$, 其中 H_l^m 是一个可高效计算的 Hash 函数类, 其中的 Hash 函数两两独立(见定义 D.1), 并将 l 比特串映射为 m 比特串, 从而得到 $S_{R,H} \in \mathcal{NP}$ 的一个实例 (x, m, h) , 使得 (x, m, h) 的正确解集合等于 $\{y \in R(x) : h(y) = 0^m\}$ 。利用 NP-证据关系 $S_{R,H}$ 到 S_R (即 S_R 的 NP-证据关系) 的简化归约 g , 可将 (x, m, h) 映射为 $g(x, m, h)$, 并有 $|\{y \in R(x) : h(y) = 0^m\}|$ 等于 $|R(g(x, m, h))|$ 。综上所述, 对于输入 x , 随机映射 M 输出实例 $M(x) \stackrel{\text{def}}{=} g(x, m, h)$, 其中 $m \in \{1, 2, \dots, l+1\}$, $h \in H_l^m$ 为均匀选择。

分析

注意: 对任意 $x \in \bar{S}_R$, 有 $\Pr[M(x) \in \bar{S}_R] = 1$ 。假设 $x \in S_R$, $m = m_x$ 的概率恰为 $1/(l+1)$, 其中 $m_x \stackrel{\text{def}}{=} \lceil \log_2 |R(x)| \rceil + 1$ 。主要讨论 $m = m_x$ 的情形, 对于均匀选择的 $h \in H_l^{m_x}$, 计算集合 $R_h(x) \stackrel{\text{def}}{=} \{y \in R(x) : h(y) = 0^{m_x}\}$ 中只有一个元素的概率下限。首先, 利用容斥原理, 得到 $\Pr_{h \in H_l^{m_x}}[|R_h(x)| > 0]$ 的下限为

$$\sum_{y \in R(x)} \Pr_{h \in H_l^{m_x}}[h(y) = 0^{m_x}] - \sum_{y_1 < y_2 \in R(x)} \Pr_{h \in H_l^{m_x}}[h(y_1) = h(y_2) = 0^{m_x}]$$

然后, 得到 $\Pr_{h \in H_l^{m_x}}[|R_h(x)| > 1]$ 的上限为

$$\sum_{y_1 < y_2 \in R(x)} \Pr_{h \in H_l^{m_x}}[h(y_1) = h(y_2) = 0^{m_x}]$$

结合这两个界限, 有

$$\begin{aligned} & \Pr_{h \in H_l^{m_x}}[|R_h(x)| = 1] \\ &= \Pr_{h \in H_l^{m_x}}[|R_h(x)| > 0] - \Pr_{h \in H_l^{m_x}}[|R_h(x)| > 1] \\ &\geq \sum_{y \in R(x)} \Pr_{h \in H_l^{m_x}}[h(y) = 0^{m_x}] - 2 \cdot \sum_{y_1 < y_2 \in R(x)} \Pr_{h \in H_l^{m_x}}[h(y_1) = h(y_2) = 0^{m_x}] \\ &= |R(x)| \cdot 2^{-m_x} - 2 \cdot \binom{|R(x)|}{2} \cdot 2^{-2m_x} \end{aligned}$$

其中最后一个等式来自于两两独立性。利用 $2^{m_x-2} < |R(x)| \leq 2^{m_x-1}$, 有

$$\Pr_{h \in H_l^{m_x}}[|R_h(x)| = 1] \geq \min_{1/4 < \rho \leq 1/2} \{\rho - \rho^2\} > \frac{1}{8}$$

因此, $\Pr[M(x) \in \text{US}_R] \geq 1/(8(l+1))$ 。定理得证。 ■

说明

定理 6.29 有时可利用具有唯一解的 SAT 来阐述。此时, 如果使用某种特殊的两两独立 Hash 函数类时, 可以避免使用简化归约。具体细节见习题 6.38。

思考

定理 6.29 的证明将两个归约步骤相结合, 其中涉及到 $S_{R,H}$ 的 NP-证据关系, 这里记作 R' 。主要步骤是搜索问题 R (或 S_R) 到求 R' (或 $(US_{R'}, \bar{S}_{R'})$) 的唯一解问题的多到一随机归约。第二个步骤是一个确定的由后一个问题到求 R 的唯一解问题的多到一映射。事实上, 定理 6.29 的证明主要是第一步, 第二步可由 R' 到 R 的简化归约提供 (由假设所保证)。根据如上说明, 在 SAT 问题中, 可以直接实现第二步。

定理 6.29 的另一种证明

注意: 对定理 6.27 证明中 (近似计数) 过程的分析暗示了, 对任意 $x \in S_R$, 存在一个整数 $i \in [l]$, 使得集合 $\{y \in R(x) : h(x) = 0^i\}$ 非空且规模为常数 (如至多有 100 个元素) 的概率 (关于 $h \in H_l^i$ 的选择) 为常数。因此, 对于均匀选择的 $i \in [l]$ 和 $h \in H_l^i$, 随机映射 $x \mapsto (x, i, h)$ 构成由 S_R 到 $S_{R,H}$ 的归约, 将任意一个“是”实例以显著概率映射为具有较少解 (如至多为 100 个) 的“是”实例。利用一个附加的随机归约 (如构造 6.32 中的归约), 可以将这种实例 (解数较少) 归约到具有唯一解的实例 (细节见习题 6.39)。

6.2.4 解的均匀生成

回顾对关系 R 的解的近似计数实际上是判定问题 S_R (区分有解与无解) 的一种变形。现在讨论一类新的计算问题, 它可以看作是搜索问题的变形。考虑这样一种计算任务, 对于一个给定实例, 生成均匀分布的解, 而不是仅仅找到一个解。然而, 正如我们将看到的, 对于许多自然问题 (以及所有的 NP-完全问题), 生成均匀分布的解可以随机归约为求一个解。

当然, 由定义可知, 能解决这个任务 (“均匀生成”) 的算法一定是随机的。我们主要讨论 $\mathcal{PC}$ 中的关系, 考虑该问题的两个版本, 其区别在于由期望的 (均匀) 分布所提供的近似程度有所不同。^①

定义 6.30 (均匀生成) 设 $R \in \mathcal{PC}$, $S_R = \{x : |R(x)| \geq 1\}$, 并设 Π 为概率过程。

(1) 称 Π 可解 R 的均匀生成问题, 如果对输入 $x \in S_R$, Π 或者输出 $R(x)$ 中一个元素, 或者输出一个特殊符号, 记作 $\perp$, 使得 $\Pr[\Pi(x) \in R(x)] \geq 1/2$, 而对任意 $y \in R(x)$, 有 $\Pr[\Pi(x) = y | \Pi(x) \in R(x)] = 1/|R(x)|$ 。

(2) 对 $\varepsilon : \mathbb{N} \rightarrow [0, 1]$, 称 Π 可解 R 的 $(1 - \varepsilon)$ -近似均匀生成问题, 如果对于输入 $x \in S_R$, $\Pi(x)$ 的分布与 $R(x)$ 的均匀分布是 $\varepsilon(|x|)$ -接近的^②。

不失一般性, 在上述两种情况中都要求当 $x \notin S_R$ 时, $\Pr[\Pi(x) = \perp] = 1$ 。更一般地, 还可要求 Π 从不输出 $R(x)$ 之外的串。

注意: 通过错误缩减, 可以使均匀生成 (第一项中) 的错误概率 (关于 $|x|$) 呈指数衰减。但相反的是, 我们并不知道是否存在一种通用的方法来减少近似均匀生成过程的偏差 (在第

① 注意: 一个严格运行于多项式时间内的概率算法不能输出某个集合上的完全均匀分布。特别地, 在标准模型下, 只允许均匀选择的二元值, 这种算法的输出不可能在势为 2 的幂的集合上是完全均匀分布的。

② 见附录 D.1.1。

二项中)。①

6.2.4.1 节中证明了对于许多搜索问题,近似的均匀生成在计算上等价于近似计数。6.2.4.2 节针对 PC 中任意搜索问题,利用 NP 预言构造了一种解均匀生成问题的直接方法。从而,任意 NP -完全问题的均匀生成问题可以随机地归约为该问题本身(或者搜索或者判定版本)。

6.2.4.1 均匀生成与近似计数的关系

我们要证明,对于 PC 中许多自然的搜索问题 R ,与 R 相关的近似计数问题在计算上等价于关于 R 的近似均匀生成问题。具体地,对于满足 $R'(x; y') \stackrel{\text{def}}{=} \{y'' : (x, y'y'') \in R\}$ 的搜索问题 $R' \in PC$,可以强简化地归约到 R ,其中由 R' 到 R 的强简化归约是指一个简化的归约 g ,伴随着一个可高效计算的一一映射,将对 $(g(x), y) \in R$ 映射为对 $(x, h(x, y)) \in R'$ (即 h 可以高效计算且 $h(x, \cdot)$ 为 $R(g(x))$ 到 $R'(x)$ 的一一映射)。出于技术上的原因,还要假设对所有 x ,有 $|g(x)| \geq |x|$ 。②注意:对于许多自然的搜索问题 R ,相应的 R' 可以强简化地归约到 R ,而附加的技术条件可以通过填充来强制性地满足(见习题 2.30)。特别地,对于搜索问题 SAT 及完美匹配问题(见习题 6.40),上述条件是成立的。我们强调以下结论对于非 NP -完全问题也成立(并且实际上,在这些问题中更有意义)。

注意:这两种类型的近似问题都有精确程度的参数,我们对上述的等价性质进行量化。

定理 6.31 设 $R \in PC$, l 为多项式,使得对任意 $(x, y) \in R$,有 $|y| \leq l(|x|)$ 。假设 R' 可以强简化地归约为 R ,其中 $R'(x; y') \stackrel{\text{def}}{=} \{y'' : (x, y'y'') \in R\}$,则有

(1) 由近似计数到近似均匀生成。令 $\varepsilon(n) = 1/5l(n)$,并令函数 $\mu: \mathbb{N} \rightarrow (0, 1)$ 满足 $\mu(n) \geq \exp(-\text{poly}(n))$,则 R 的 $(1-\mu)$ -近似均匀生成可以在概率多项式时间内归约到 $\#R$ 的 $(1-\varepsilon)$ -近似计数。

(2) 由近似均匀生成到近似计数。对任意非增且不可忽略的 $\varepsilon: \mathbb{N} \rightarrow (0, 1)$ (即对任意 n , $\varepsilon(n) \geq 1/\text{poly}(n)$), $\#R$ 的 $(1-\varepsilon)$ 近似计数可在概率多项式时间内归约为 R 的 $(1-\varepsilon')$ -近似均匀生成,其中 $\varepsilon'(n) = \varepsilon(n)/7l(n)$ 。

实际上,当 R' 只能简化地归约为 R 时,第一部分也成立。

注意:在第一部分中近似均匀生成的近似程度(即 μ)独立于所归约到的计数问题的近似程度(即 ε),条件是近似计数器的性能优于某个阈值。另一方面,第二部分中计数器的近似程度(即 ε)却依赖于近似均匀生成的近似程度(即 ε'),但这种依赖不会超过某个界限(即明显地相对偏差)。回顾对于简化归约下的 NP -完全问题,近似计数的程度是可以提高的(见习题 6.34)。然而,对于非 NP -完全问题,定理 6.31 十分有用。③因为对于 NP -完全问题而言,近似计数和近似均匀生成均可以随机归约为相应的搜索问题(见习题 6.42)。

证明:在整个证明中,为了更加简洁,假设(事实上不失一般性) $R(x) \neq \emptyset$ 且 $R(x) \subseteq \{0, 1\}^{l(|x|)}$ 。

① 注意:在某些情况下,近似均匀生成算法的偏差是可以减少的,见定理 6.31 之后的讨论。

② 这个技术性条件允许我们将用 $|g(x)|$ 表示的偏差上界替换为用 $|x|$ 表示的偏差上界,这依赖于一个事实,即对任意非增的函数 $\varepsilon: \mathbb{N} \rightarrow (0, 1)$,有 $\varepsilon(|g(x)|) \leq \varepsilon(|x|)$ 。

③ 事实上,许多近似计数问题都明确或隐含地依赖于定理 6.31 (见参考文献[168]中 11.3.1 节及参考文献[131])。

对第一部分, 首先将 R 的均匀生成问题归约为 $\#R$ (而不是求 $\#R$ 的近似值)。对于输入 $x \in S_R$, 逐比特地随机生成均匀分布的 $y \in R(x)$ 。下面进行迭代, 第 i 次迭代的输入是一个 $(i-1)$ 比特长的串 y' , 满足 $R'(x; y') \stackrel{\text{def}}{=} \{y'' : (x, y'y'') \in R\}$ 非空。以概率 $|R'(x; y'1)|/|R'(x; y')|$ 第 i 比特为 1, 以概率 $1 - |R'(x; y'1)|/|R'(x; y')|$ 置其为 0。可以利用 $R' = \{(x; y'), y'' : (x, y'y'') \in R\} \in PC$ 到 R 的简化归约 g 来得到 $|R'(x; y'1)|$ 和 $|R'(x; y')|$ 。也就是说, 通过询问 $|R(g(x; y'))|$ 的值来得到 $|R'(x; y')|$ 。忽略完整性问题, 只要能访问预言 $\#R$, 则所有这些工作都是完善的 (即我们生成了一个在 $R(x)$ 上均匀分布的 $l(n)$ -比特串)。但是由于只能访问 $\#R$ 的近似预言, 所以在实现上述思想时需要格外仔细。

将近似预言记作 A , 首先, 由充分的错误缩减, 我们可以假设对任意 z , 有 $\Pr[A(z) \in (1 \pm \varepsilon(n)) \cdot \#R(z)] > 1 - \mu'(|z|)$, 其中 $\mu'(n) = \mu(n)/l(n)$ 。在剩余的分析中, 将忽略由随机预言 A (对于询问 z) 提供的对 $\#R(z)$ 的估值偏离上述区间的概率 (注意: 这种小概率事件是导致输出分布与 $R(x)$ 上均匀分布不一致的唯一根源)。^①下面暂时假设 A 是确定的, 且对任意的 x 和 y' , 有

$$A(g(x, y'0)) + A(g(x, y'1)) \leq A(g(x, y')) \quad (6.10)$$

还假设近似在“平凡程度上” (其中只需检查是否 (x, y) 属于 R) 是正确的, 即对任意 $y \in \{0, 1\}^{l(|x|)}$, 有

$$\begin{cases} A(g(x; y)) = 1, & (x, y) \in R \\ A(g(x; y)) = 0, & \text{其他} \end{cases} \quad (6.11)$$

对上述过程中的第 i 次迭代进行修改, 使得当输入为 $(i-1)$ -比特长的前缀 y' 时, 依概率 $A(g(x; y'\sigma))/A(g(x; y'))$ 置第 i 比特为 $\sigma \in \{0, 1\}$, 并依剩余的概率 (即 $1 - (A(g(x; y'0))/A(g(x; y')) - (A(g(x; y'1))/A(g(x; y'))))$) 停机 (输出 $\perp$)。事实上, 式 (6.10) 保证了后一个要求是合理的, 因为两个主要概率之和至多为 1。如果完成了最后一次迭代 (即第 $l(|x|)$ 次), 则输出生成的 $l(|x|)$ -比特串。因此, 只要式 (6.10) 成立 (不考虑近似量化其他方面的问题), 则输出每个 $y = \sigma_1 \sigma_2 \cdots \sigma_{l(|x|)} \in R(x)$ 的概率为

$$\frac{A(g(x; \sigma_1))}{A(g(x; \lambda))} \cdot \frac{A(g(x; \sigma_1 \sigma_2))}{A(g(x; \sigma_1))} \cdots \frac{A(g(x; \sigma_1 \sigma_2 \cdots \sigma_{l(|x|)}))}{A(g(x; \sigma_1 \sigma_2 \cdots \sigma_{l(|x|-1)})} \quad (6.12)$$

由式 (6.11) 可知, 上述概率等于 $1/A(g(x; \lambda))$ 。因此, 上述过程等概地输出 $R(x)$ 中的每个元素, 从不输出除 $\perp$ 之外的不属于 $R(x)$ 的值。从而, 近似的程度只影响输出非 $\perp$ 值的概率 (这又等于 $|R(x)|/A(g(x; \lambda))$)。关键在于, 只要式 (6.11) 成立, 则由上述程序得到的特定近似值并不重要——除 $A(g(x; \lambda))$ 外, 所有这些值都可以“删除”。

现在转向式 (6.10) 和式 (6.11) 的实现。在实现式 (6.11) 时, 可以直接检查 (是否 $(x, y) \in R$)

① 注意: 这些小概率事件造成的 (可忽略的) 影响不容易纠正。对于初学者, 我们不一定指出这类事件何时发生。另外, 这类事件可能在算法的不同调用中 (即对于不同询问) 以不同概率发生。

而不是调用 $A(g(x, y))$ 。^①至于式(6.10), 要人为地利用 $A'(x, y') \stackrel{\text{def}}{=} (1 + \varepsilon(|x|))^{3(l(|x|) - |y'|)} \cdot A(g(x, y'))$ 而非 $A(g(x, y'))$ 来实现。联想到 $A(g(x, y')) = (1 \pm \varepsilon(|x|)) \cdot |R'(x, y')|$, 有

$$A'(x, y') > (1 + \varepsilon(|x|))^{3(l(|x|) - |y'|)} \cdot (1 - \varepsilon(|x|)) \cdot |R'(x, y')|$$

$$A'(x, y'\sigma) < (1 + \varepsilon(|x|))^{3(l(|x|) - |y'| - 1)} \cdot (1 - \varepsilon(|x|)) \cdot |R'(x, y'\sigma)|$$

利用 $(1 - \varepsilon(|x|)) \cdot (1 + \varepsilon(|x|))^3 > (1 + \varepsilon(|x|))$ 可证明断言(式(6.10))成立。注意: 上述修改只影响输出非 $\perp$ 值的概率, 而这个良好事件发生的概率为 $|R'(x, \lambda)| / A'(x, \lambda)$, 其下限是 $(1 + \varepsilon(|x|))^{-3(l(|x|) + 1)} > 1/2$, 这里的不等式与 ε 的设置(即 $\varepsilon(n) = 1/5l(n)$)有关。最后, 讨论 A 为确定的假设。该假设只用于识别在第 $(|y'| - 1)$ 次迭代中得到并使用的 $A(g(x, y'))$ 值, 以及第 $|y'|$ 次迭代中得到并使用的 $A(g(x, y'))$ 值。为了达到同样的效果, 也可以(在第 $|y'|$ 次迭代中)重复使用前一个值而不是再次调用 A 。第一部分得证。

为证明第二部分, 首先要将求 $\#R$ 近似值的任务归约为(精确地)均匀生成 R 。对于输入的 $x \in S_R$, 归约中以不同方式使用 $R(x)$ 中元素的可能前缀所构成的树。再次使用迭代。第 i 次迭代的输入是 $i-1$ 比特串 y' , 满足 $R'(x, y') \stackrel{\text{def}}{=} \{y'' : (x, y'y'') \in R\}$ 非空。在第 i 次迭代中, 通过在 $R'(x, y')$ 上的均匀分布进行均匀采样来估计两个分数 $|R'(x, y'0)| / |R'(x, y')|$ 和 $|R'(x, y'1)| / |R'(x, y')|$ 中哪个较大。即在 $R'(x, y')$ 上的均匀分布中采样 $\text{poly}(|x|/\varepsilon'(|x|))$ 次, 以极高概率得到对两个分数的估值, 其加法偏差至多为 $\varepsilon'(|x|)$ 。这意味着我们得到了对大于 $1/3$ 的上述两个分数之一(或两者均大于 $1/3$)的近似因子为 $1 \pm 3\varepsilon'(|x|)$ 的相关近似。事实上, 对另一个分数(当其较小时), 可能不会得到这样好的相关近似, 但这并不重要。如果无法识别两个分数中哪个较大也没关系, 只要使用的近似能指示一个量(如大于 $1/3$)即可。在下次迭代中, 向 y' 中添加对应于这个量的比特。具体而言, 假设我们得到了近似值 $a_0(y') \approx |R'(x, y'0)| / |R'(x, y')|$ 及 $a_1(y') \approx |R'(x, y'1)| / |R'(x, y')|$, 如果 $a_1(y') > a_0(y')$, 则向 y' 中添加比特 1, 否则, 添加 0。最后, 当得到的 $y = \sigma_1 \sigma_2 \cdots \sigma_{l(|x|)}$ 满足 $(x, y) \in R$ 时, 输出

$$a_{\sigma_1}(\lambda)^{-1} \cdot a_{\sigma_2}(\sigma_1)^{-1} \cdots a_{\sigma_{l(|x|)}}(\sigma_1 \sigma_2 \cdots \sigma_{l(|x|)-1})^{-1} \quad (6.13)$$

其中对每个 i , 有 $a_{\sigma_i}(\sigma_1 \sigma_2 \cdots \sigma_{i-1}) = (1 \pm 3\varepsilon'(|x|)) \cdot \frac{|R'(x, \sigma_1 \sigma_2 \cdots \sigma_i)|}{|R'(x, \sigma_1 \sigma_2 \cdots \sigma_{i-1})|}$ 。

与第一部分情况类似, 关于 R' 的操作(此时是对 R' 均匀生成解)可以通过到 R 的简化归约 g 来实现。即无论何时, 当需要在集合 $R'(x, y')$ 中均匀采样时, 先在集合 $R(g(x, y'))$ 中采样, 并利用 g 为强简化归约这一假设所保证的映射来恢复 $R'(x, y')$ 中相应的元素。最后, 注

① 另外, 注意到, 由于 A 是一个 $(1 - \varepsilon)$, $\varepsilon > 1$, 则 $\#R'(z) = 0$ 必定蕴含着 $A(z) = 0$ 。还有, 由于 $\varepsilon < 1/3$, 如果 $\#R'(z) = 1$, 则 $A(z) \in (2/3, 4/3)$, 可以四舍五入为 1。

意到, 目前为止只假设了对 R 的均匀生成, 而使用一个 $(1-\varepsilon')$ -近似均匀生成只意味着所有的近似偏差增加了 ε' 。因此, 对每个 i , 有

$$a_{\sigma_i}(\sigma_1\sigma_2\cdots\sigma_{i-1}) = (1 \pm 6\varepsilon'(|x|)) \cdot \frac{|R'(x; \sigma_1\sigma_2\cdots\sigma_i)|}{|R'(x; \sigma_1\sigma_2\cdots\sigma_{i-1})|}$$

以极高概率成立。从而, 对于输入 x , 当使用能提供 R 的 $(1-\varepsilon')$ -近似均匀生成预言时, 输出属于

$$\prod_{i=1}^{l(|x|)} \left((1 \pm 6\varepsilon'(|x|))^{-1} \cdot \frac{|R'(x; \sigma_1\sigma_2\cdots\sigma_{i-1})|}{|R'(x; \sigma_1\sigma_2\cdots\sigma_i)|} \right) \quad (6.14)$$

的概率极高。其中的错误概率是由于在某次迭代中, 近似值与正确值的差大于加法偏差 $2\varepsilon'(n)$, 而这个事件几乎不可能发生。注意到, 式 (6.14) 等于 $(1 \pm 6\varepsilon'(|x|))^{-l(|x|)} \cdot |R(x)|$, 利用 $(1 \pm 6\varepsilon'(|x|))^{-l(|x|)} \subset (1 \pm \varepsilon(|x|))$ (对 $\varepsilon' = \varepsilon/7l$ 成立), 第二部分得证。■

6.2.4.2 一个直接的均匀生成程序

在本章的最后, 我们构造一个程序, 可以直接求解任意 $R \in \mathcal{PC}$ 的均匀生成问题。程序不可避免地使用一个 $\mathcal{NP}$ 预言 (或者对于 $\mathcal{NP}$ -完全问题, 直接调用预言 S_R), 因为求解 R 的均匀生成蕴含着求解相应的搜索问题 (这又蕴含着 S_R 的成员判定)。该程序的一个优点是, 在 6.2.4.1 节中的归约下, 解决了均匀生成问题而不是近似均匀生成问题。

我们将再次使用 Hash 函数, 但此次使用的 Hash 函数类具有更强的“均匀性质” (见附录 D.2.3)。具体地, 使用 l 元独立的 Hash 函数类, 将 l 比特串映射为 m 比特串, 其中 l 是 R 的解的长度上限, 而且要依赖一个事实, 即这个 Hash 函数类满足引理 D.6。直观上, 这种函数将 $\{0,1\}^l$ 分割为 2^m 份, 而引理 D.6 中断言, 这样可以“均匀分割”所有足够大的集合。也就是说, 对任意规模为 $\Omega(l \cdot 2^m)$ 的集合 $S \subseteq \{0,1\}^l$, 由该函数类中几乎每个函数所诱导的分割都可以使分割后的每一份包含 S 中约 $|S|/2^m$ 个元素。特别地, 如果 $|S| = \Theta(l \cdot 2^m)$, 则每一份包含 S 中的 $\Theta(l)$ 个元素。将这个函数类记作 H_l^m , 并要求其中元素具有简单而有效的表示 (如附录 D.2.1 中的定义)。

不严格地说, 以下的 (均匀生成) 过程首先随机选择一个 Hash 函数, 并检查其是否能“均匀分割”目标集 $S = R(x)$ 。如果这个条件满足, 则程序随机选择分割后的一份, 并提取其中的所有元素。最后, 程序或者输出某个元素, 或者停机且不输出任何值, 而输出每一个元素的概率 p 是固定的 (与被选中划分中的元素个数无关)。这保证了输出每个元素 $e \in S$ 的概率相同 (即 $2^{-m} \cdot p$), 而无需考虑 S 中与 e 位于同一份中的元素个数。

在以下构造中, 我们假设对于输入 x , 还能得到关于 $R(x)$ 规模的良好估值。为实现这个假设, 可以在预处理阶段使用一个近似计数程序。或者还可以说, 以下构造中的思想实际上可以得到一个近似计数程序。

构造 6.32 (均匀生成) 对于输入 x 和 $m'_x \in \{m_x, m_x + 1\}$, 其中 $m_x \stackrel{\text{def}}{=} \lfloor \log_2 |R(x)| \rfloor$, 且 $R(x) \in \{0,1\}^l$, 预言机进行如下操作。

(1) 选择“均匀分割” $R(x)$ 后的一份。机器设置 $m = \max(0, m'_x - \log_2 40l)$ 并均匀选择 $h \in H_l^m$ 。该函数确定了将 $\{0,1\}^l$ 分割为 2^m 份的一种划分^①，且我们希望每一份包含的 $R(x)$ 中元素个数几乎相同。下面机器检查这个条件是否满足，或者没有任何一份的元素个数超过 $120l$ （即多于期望元素个数的两倍）。实现方法是检查 (x, h, l^{120l+1}) 是否属于集合 $S_{R,H}^{(1)}$ ，该集合定义如下：

$$\begin{aligned} S_{R,H}^{(1)} &\stackrel{\text{def}}{=} \left\{ (x', h', l') : \exists v, \text{ 使得 } \left| \{ y : (x', y) \in R \wedge h'(y) = v \} \right| \geq l \right\} \\ &= \left\{ (x', h', l') : \exists v, y_1, y_2, \dots, y_l, \text{ 使得 } \psi^{(1)}(x', h', v, y_1, y_2, \dots, y_l) \right\} \end{aligned} \quad (6.15)$$

其中 $\psi^{(1)}(x', h', v, y_1, y_2, \dots, y_l)$ 成立当且仅当 $y_1 < y_2 < \dots < y_l$ 且对任意 $j \in [l]$ ，有 $(x', y_j) \in R \wedge h'(y_j) = v$ ，注意： $S_{R,H}^{(1)} \in \mathcal{NP}$ 。

如果上述检查的回答是肯定的（即存在一个划分，其中包含了多于 $120l$ 个元素），则停机并输出 $\perp$ ，否则，机器利用所选择的 h 继续运行。此时，没有任何一个划分中包含多于 $120l$ 个元素（即对任意 $v \in \{0,1\}^m$ ，有 $\left| \{ y : (x, y) \in R \wedge h(y) = v \} \right| \leq 120l$ ）。我们强调这个条件是绝对可以保证的，这是由于 $(x, h, l^{120l+1}) \notin S_{R,H}^{(1)}$ 。

(2) 选择分割后的一份并确定 $R(x)$ 中有多少元素包含在其中。机器均匀选择 $v \in \{0,1\}^m$ 并通过发出向如下 NP-集 $S_{R,H}^{(2)}$ 的询问来确定 $s_v \stackrel{\text{def}}{=} \left| \{ y : (x, y) \in R \wedge h(y) = v \} \right|$ ，即

$$S_{R,H}^{(2)} \stackrel{\text{def}}{=} \left\{ (x', h', v', l') : \exists y_1, y_2, \dots, y_l \text{ 使得 } \psi^{(1)}(x', h', v', y_1, y_2, \dots, y_l) \right\} \quad (6.16)$$

具体地，对 $i=1,2,\dots,120l$ ，机器检查 (x, h, v, l^i) 是否属于 $S_{R,H}^{(2)}$ ，并置 s_v 为得到肯定回答的最大 i 值。

(3) 提取选中划分中的所有元素，并随机输出其中之一。程序利用 s_v ，并通过向如下的 NP-集 $S_{R,H}^{(3)}$ 发出询问来重构集合 $S_v \stackrel{\text{def}}{=} \{ y : (x, y) \in R \wedge h(y) = v \}$ ，即

$$S_{R,H}^{(3)} \stackrel{\text{def}}{=} \left\{ (x', h', v', l', j) : \exists y_1, y_2, \dots, y_l \text{ 使得 } \psi^{(3)}(x', h', v', y_1, y_2, \dots, y_l, j) \right\} \quad (6.17)$$

其中 $\psi^{(3)}(x', h', v', y_1, y_2, \dots, y_l, j)$ 成立当且仅当 $\psi^{(1)}(x', h', v', y_1, y_2, \dots, y_l)$ 成立，且 $y_1, y_2, \dots, y_l$ 中第 j 比特为1。具体地，对于 $j_1=1,2,\dots,s_v$ 及 $j_2=1,2,\dots,l$ ，发出询问 $(x, h, v, l^{s_v}, (j_1-1) \cdot l + j_2)$ 来确定 y_{j_1} 的第 j_2 个比特。最后，重构 S_v 之后，程序以概率 $1/120l$ 输出每个 $y \in S_v$ ，否则，输出 $\perp$ （即以概率 $1 - (s_v/120l)$ ）。

注意：对于 $|R(x)| = \Omega(l)$ 以及 $m = m'_x - \log_2 40l$ ，引理 D.6 蕴含了以绝对的高概率（对于 $h \in H_l^m$ 的选择）有：每个集合 $\{ y : (x, y) \in R \wedge h(y) = v \}$ 的势为 $(1 \pm 0.5)|R(x)|/2^m$ 。因此，忽略 $|R(x)| = O(l)$ 的情况，利用第一步可得到对 $\#R$ 的近似计数程序，见习题 6.41，利用第二部的

① 为了满足均匀性，还允许 $m=0$ 的情形，这可以人为规定。此时， H_l^0 中的所有 Hash 函数将 $\{0,1\}^l$ 映射为空串，记作 0^0 ，从而定义了对 $\{0,1\}^l$ 的一种平凡分割（即分成一份）。

思想, 该程序还可处理 $|R(x)| = O(l)$ 的情形。然而, 我们的目标是证明如下结论。

命题 6.33 构造 6.32 解决了 R 的均匀生成问题。

证明: 直观上, 由引理 D.6 (以及 m 的设置), 均匀选择的 $h \in H_l^m$ 以极高概率将 $R(x)$ 划分为 2^m 份, 每份中元素个数不超过 $120l$ 。以下对这一事实给出一个冗长的证明。由于 $m = \max(0, m'_x - \log_2 40l)$, 主要考虑 $m'_x > \log_2 40l$ 的情形 (其他情况下, $|R(x)| \leq 2^{m'_x+1} \leq 80l$)。此时, 由引理 D.6 (利用 $\varepsilon = 0.5$ 以及 $m = m'_x - \log_2 40l \leq \log_2 |R(x)| - \log_2 20l$ (这蕴含着 $m \leq \log_2 |R(x)| - \log_2 (5l/\varepsilon^2)$)), 每个集合 $\{y: (x, y) \in R \wedge h(y) = v\}$ 的势为 $(1 \pm 0.5)|R(x)|/2^m$ 的概率极高。利用 $m'_x > (\log_2 |R(x)|) - 1$ (以及 $m = m'_x - \log_2 40l$), 有 $|R(x)|/2^m < 80l$, 从而每一份中至多包含 $R(x)$ 中的 $120l$ 个元素。还要注意的, 利用 $m'_x \leq (\log_2 |R(x)|) + 1$, 有 $|R(x)|/2^m \geq 20l$, 从而每一份中至少包含 $10l$ 个元素。

对第一步的一个重要观察是, 如果程序在第一步不停机, 则实际上在 h 诱导的 $R(x)$ 的划分中, 每一份至多含 $120l$ 个元素。这些子集中元素个数可能不同, 但这并不重要, 因为输出每个元素的概率都相同 (即 $1/120l$)。重要的是, 不同子集的平均元素个数要足够大, 因为这个平均数决定了程序输出 $R(x)$ 中元素 (而非 $\perp$) 的概率。具体地, 如果第一步不停机, 则第三步输出 $R(x)$ 中某个元素的概率等于各个子集的平均元素 (即 $|R(x)|/2^m$) 个数除以 $120l$ 。回顾对于 $m > 0$ (或 $m = 0$), 有 $|R(x)|/2^m \geq 20l$ (或 $|R(x)| \geq 1$), 在这种情形下, 输出 $R(x)$ 中某个元素的概率至少为 $1/6$ (或 $|R(x)|/120l$)。前面提到, 算法在第一步停机的概率是可以忽略的, 从而程序输出 $R(x)$ 中某个元素的概率至少为 $0.99 \cdot \min(|R(x)|/120l, 1/6)$ 。 ■

说明

构造 6.32 的性能可以很容易地得到改善, 只需将 $m = 0$ 的情形单独处理即可。此时, 第 3 步可以简化, 并通过均匀选择及输出 S_λ (等于 $R(x)$) 中一个元素来改善。在这样修改之后, 程序输出 $R(x)$ 中某个元素的概率至少为 $1/6$ 。任何情况下, 均匀生成程序输出 $\perp$ 的概率都可通过重复调用来减小。

思考

构造 6.32 是 “Hash 范式” 应用的巅峰。该范式的目的是允许对任意集合进行各种操作。特别是, 如构造 6.32 所示, 可以将一个较大的集合分割成较小的子集, 且这些子集中的元素个数几乎相同。我们强调实现这样的 Hash 函数时, 是在一个充分的类中随机选择一个函数。事实上, 随机化的这种用途 (即允许对较大集合的操作) 似乎责无旁贷。

本章注释

随机程序一个重要的方面是其成功概率, 这显然是一个量化概念, 它在 6.1 节中的概率多项式算法与 6.2 节中的计数问题间建立起清晰的联系 (还可见于习题 6.20)。6.2 节给出了随机程序与计数问题间一些更有趣的联系 (如随机化在近似计数中的应用)。由于有这些联系, 我们在同一章讨论这两个问题。

随机算法

让人们改变习惯需要充分的理由,事实上,对随机算法的研究动机来自于几个著名例子。出乎意料的是,两个最著名的例子(下面将讨论)由于后来的发现,已经有点被人们淡忘了,但是先抛开其地位不谈,在这两个例子的研究中涌现出的基本问题仍很吸引人。这些问题主要涉及到随机化在各种计算环境中的功能,特别是在判定问题和搜索问题当中。在简要介绍两个例子之后,我们将回到这些问题。

第一个例子:素性检测

在超过 20 年的时间里,素性检测是说明随机化在高效算法中起作用的一个典型例子。著名的 Solovay 和 Strassen 算法^[211]以及 Rabin 算法^[184]出现于 20 世纪 70 年代末,这两个算法表明,素数判定属于 coRP 类(即这些测试总是能识别正确的素数,但当输入为合数时有可能误判)(构造 6.4 中使用的方法由 M.Blum 率先提出,它只说明素数判定属于 BPP)。20 世纪 80 年代末,Adleman 和 Huang^[2]证明了素数判定属于 RP (从而也属于 ZPP)。21 世纪初 Agrawal、Kayal 和 Saxena^[3]证明了素数的判定实际上属于 P 类。然而,我们应该注意的是,事实上,从一开始就有强大证据表明素数判定属于 P 类:可以参考 Miller 的确定算法^[166],它依赖于扩展的 Riemann 假设。

第二个例子:无向连通性

用以说明随机化功能的另一个著名例子是检测图的无向连通性,特别是在对数空间计算中。构造 6.12 中的随机漫游算法由 Aleliunas、Karp、Lipton、Lovász 和 Rackoff^[5]提出。注意:第一个确定的对数空间算法直到 25 年后才被提出(见 5.2.4 节或参考文献[190])。

另一个著名例子:多项式相等的测试

与前两个例子出现于同一时期的第三个著名例子是多项式相等的测试^[65,199,243]。6.1.3.1 节中提出的测试算法在复杂性理论中有许多应用(后续章节中介绍了一些应用)。毫无疑问,在构造 6.7 的抽象环境中,随机化是必不可少的。有趣的是,习题 6.17 中提到的计算版本至今仍被看作是抗拒去随机化的尝试(见参考文献[134])。

其他随机算法

除了上述 3 个例子,还有其他几个著名的随机算法。在搜索和判定问题范围内,这类算法包括求图的完美匹配和最小割集(如参考文献[90]中附录 B.1 或参考文献[168]中 12.4 节及 10.2 节),并注意到随机化在计算数论中有突出贡献(如参考文献[24]或参考文献[168]中第 14 章)。我们指出,近似问题中大量应用了随机算法,特别是近似计数问题(如参考文献[168]中第 11 章)。推荐有兴趣的读者参考随机算法方面的通用教科书,即参考文献[168]。

虽然可以证明,在几个重要计算环境中随机化非常关键(如第 9 章,10.1.2 节,附录 C 以及附录 D.3),但一个基本问题是在搜索和判定问题中,随机化是不是必需的。流行的假设是随机化在时间受限和空间受限的算法中作用不大。例如,人们假设 $\text{BPP}=\text{P}$ 和 $\text{BPL}=\text{L}$ 。注意:这些假设并不排除随机化确实起作用的情况,而只表明其作用有限。例如,任意一个平方时间的随机算法可以用立方时间的确定算法来模仿,而不是用一个平方时间的确定算法。

对 BPP 类的研究

$\text{BPP}=\text{P}$ 的假设被看作是对 BPP 类的完全去随机化,可以证明,在某些合理的困难性

假设下它也成立。8.3 节将给出这个结论（以及相关的其他结论）。本章只提出关于 BPP 类一些无条件的结论，如 $BPP \subset P/poly$ 以及 $BPP \subseteq PH$ 。我们对定理 6.9 的证明采用了 Lautemann 的证明思想^[150]。Sipser（独立地）给出了另一种不同的证明技巧^[207]，得到的结论较弱但是有更多应用背景（见定理 6.27 和 F.2）。

承诺问题的作用

除了用于定理 6.9，承诺问题还可用于证明完全问题和随机算法的层级定理（分别见习题 6.14 和习题 6.15）。我们指出，在相应的标准判定问题类中这些结论是否成立尚且未知。技术上的难点在于无法模仿或/和识别使用非平凡概率判定规则的概率机。

随机运算的可行性

关于这一问题，在第 8 章和附录 D.4 给出了不同观点。具体来讲，在第 8 章，实现随机算法时并不一定要生成均匀分布的比特序列，只要生成的序列（对使用者而言）看上去是均匀分布就足够了。在许多情况下，这个弱化的要求可以减少实现中所需的随机数个数，甚至可以导致完全（高效的）去随机化（见 8.3 节和 8.4 节）。附录 D.4 中提出了一种不那么极端的方法，它考虑从弱的随机源中提取几乎均匀分布的比特序列。不用说，这两种方法是互补的，并且可以结合使用。

计数问题

计数类 $\#P$ 最早由 Valiant 提出^[230]，他证明了计算 0/1 矩阵的积和式是一个 $\#P$ -完全问题（即定理 6.20）。有趣的是，与 Cook 对 NP-完全问题的引入相似^[58]，Valiant 的动机也是确定某个具体问题的复杂性（即求矩阵积和式）。

我们在证明定理 6.20 时是基于 Valiant 的论文^[230]和其后的一些研究（最著名的是参考文献[31]）。具体地，命题 6.21 中归约的高结构性以及对子句的“结构性”构造来自于^[230]，而图 6.3 中起重要作用的分图则基于参考文献[31]。命题 6.22 的证明也基于参考文献[31]（但对其进行了一些变形）。至于子句构造问题，我们很遗憾没能引用和/或使用这一设计问题的系统化研究。

在正文中还提到，我们准备给出 Toda 定理^[220]的证明，其中断言 PH 中每个集合都可以 Cook-归约到 $\#P$ （即定理 6.16）。附录 F.1 中有一个相关结论的证明，其中暗示 PH 可以在概率多项式时间内归约为 $\#P$ 。其他证明还可见于参考文献[136, 212, 220]。

近似计数及相关问题

解 $\#P$ 的近似程序由 Stockmeyer^[214]基于 Sipser 的思想^[207]而提出。我们的陈述则参考了该领域中更新的发展。 NP 到唯一解问题的随机归约由 Valiant 和 Vazirani 发现^[233]。我们对此的阐述仍有所不同。

近似计数与均匀生成间的关系（见 6.2.4.1 节）由 Jerrum、Valiant 和 Vazirani^[132]发现，最后这个结论在算法设计中发挥了极大作用（如“马尔可夫链方法”（见参考文献[168]中 11.3.1 节））。均匀生成的直接算法（见 6.2.4.2 节）来自参考文献[28]。

6.2.2.1 节的内容建立在参考文献[140]的基础上，建议感兴趣的读者见参考文献[131]，其中提出了一个概率多项式时间算法，用于估算非负矩阵的积和式。这个诱人的算法基于一个事实：如果矩阵 M 与 M' 足够相似，则（近似地）知道了非负矩阵 M 的某些参数之后，可以近似求得关于矩阵 M' 的同样参数。具体地， M 和 M' 可以只有一项不同，且相应值的比率必

须接近 1。当然,实际上会考虑(不是普遍适用,但更恰当)矩阵的特定参数,包括其积和式。因此,给定一个矩阵 M , 要求其积和式的近似值, 考虑矩阵序列 $M_0, M_1, \dots, M_t \approx M$, 其中 M_0 为全 1 矩阵(其积和式易求), 每个 M_{i+1} 由 M_i 求得, 为 M_i 的某一项乘以一个接近 1 的因子而将其减小, 从而得到 M_{i+1} 。通过这个(多项式次)渐变过程可以将 M_0 修改为非常接近 M 的矩阵 M_t (从而其积和式也非常接近 M 的积和式)。因此, 由 M_t 积和式的近似值可以估算 M 的积和式。

最后, 我们指出, 10.1.1 节提供了对不同类型近似问题的处理方法。具体地, 当给定(搜索问题 R 的)一个实例 x 时, 不是求解的数量(即 $\#R(x)$) 的近似值, 而是求最佳解的近似值(即最佳的 $y \in R(x)$), 其中解的取值由一个辅助函数定义。

习 题

6.1 证明如果一个搜索(或判定)问题可以由概率多项式时间算法求解, 且算法的失效概率为 0, 则该问题可由一个确定的多项式时间算法求解。

提示: 将算法的内部随机数用一个易于确定生成的固定结果代替(如全 0 序列)。

6.2 (随机归约) 续 6.1 节的讨论, 证明如下结论。

(1) 如果问题 Π 可在多项式时间内归约为一个可用概率多项式时间算法求解的问题, 则 Π 也可在概率多项式时间内求解, 这里的求解是指以可忽略的错误概率求解。

警告: 当 Π' 为搜索问题时, 要求对于输入 x , 求解算法能以至少 $1 - \mu(|x|)$ 的概率输出一个正确解, 而不要求总是输出同一个解。

提示: 不失一般性, 归约不会两次发出同一询问。

(2) 证明概率多项式时间归约具有传递性。

(3) 证明随机的 Karp-归约具有传递性, 并可以从中得到概率多项式时间归约的特例。

定义单边错误及零边错误的随机(Karp 和 Cook)归约, 并考虑将上述三项应用于这些归约。注意: 单边错误的情形有些微妙。

6.3 (概率求解搜索问题的定义) 续 6.1.2 节一开始的讨论, 假设对某个概率多项式时间算法 A 及正多项式 p , 有以下结论成立: 对任意 $x \in S_R \stackrel{\text{def}}{=} \{z: R(z) \neq \emptyset\}$, 存在 $y \in R(x)$ 使得 $\Pr[A(x) = y] > 0.5 + (1/p(|x|))$, 而对任意 $x \notin S_R$, 有 $\Pr[A(x) = \perp] > 0.5 + (1/p(|x|))$ 。

(1) 证明存在一个概率多项式时间算法, 能以可忽略的错误概率解搜索问题 R 。

提示: 见习题 6.4 中的相关算法。

(2) 考虑要求一个(正确)解以超过 $0.5 + (1/p(|x|))$ 的概率出现的必要性。具体地, 如果只保证对任意 $x \in S_R$, 有 $\Pr[A(x) \in R(x)] > 0.5 + (1/p(|x|))$ (而对任意 $x \notin S_R$, 有 $\Pr[A(x) = \perp] > 0.5 + (1/p(|x|))$), 则情况会怎样呢?

注意: 并不一定要求 R 属于 PC , 事实上, 当 $R \in PC$ 时, 可以对每个 $x \notin S_R$ 减少错误概率, 并对 $x \in S_R$ 进行错误缩减, 正如 RP 的情形。

6.4 (BPP 类的错误缩减) 对 $\varepsilon: \mathbb{N} \rightarrow [0, 1]$, 定义 BPP_ε 为可在概率多项式时间内以错误上限 ε 求解的判定问题类。证明以下两个断言。

(1) 对每个正多项式 p 及 $\varepsilon(n) = (1/2) - (1/p(n))$, BPP_ε 类等于 BPP 类。

(2) 对每个正多项式 p 及 $\varepsilon(n) = 2^{-p(n)}$, BPP 类等于 BPP_ε 类。

对搜索问题给出类似的定义。具体地, 对每个有解的输入, 考虑输出一个特定解的概率。

提示: 给定一个语义上较弱的算法 A , 考虑一个算法 A' , 对于输入 x , 调用 A 算法 $t(|x|)$ 次, 并利用多数原则。对第一部分, 令 $t(n) = O(p(n)^2)$, 并应用 Chebyshev 不等式。对第二部分, 令 $t(n) = O(p(n))$, 并应用 Chernoff 界。

6.5 (PR 类的错误缩减) 对 $\rho: \mathbb{N} \rightarrow [0, 1]$, 定义判定问题类 RP_ρ : 如果存在概率多项式时间算法 A , 使得对任意 $x \in S$, 有 $\Pr[A(x)=1] \geq \rho(|x|)$, 而对任意 $x \notin S$, 有 $\Pr[A(x)=0]=1$, 则 S 属于 RP_ρ 类。证明以下两个断言。

(1) 对每个正多项式 p , $RP_{1/p}$ 类等于 RP 类。

(2) 对每个正多项式 p , RP 类等于 RP_ρ 类, 其中 $\rho(n) = 1 - 2^{-p(n)}$ 。

提示: 单边错误允许使用“或原则”(而不是“多数原则”)进行判定。

6.6 (ZPP 类的错误缩减) 对于 $\rho: \mathbb{N} \rightarrow [0, 1]$, 定义判定问题类 ZPP_ρ : 如果存在概率多项式时间算法 A , 使得对任意 x , 有 $\Pr[A(x) = \chi_S(x)] \geq \rho(|x|)$ 且 $\Pr[A(x) \in \{\chi_S(x), \perp\}] = 1$, 其中当 $x \in S$ 时, $\chi_S(x) = 1$, 否则, $\chi_S(x) = 0$, 则 S 属于 ZPP_ρ 类。证明以下两个断言。

(1) 对每个正多项式 p , $ZPP_{1/p}$ 类等于 ZPP 类。

(2) 对每个正多项式 p , ZPP 类等于 ZPP_ρ 类, 其中 $\rho(n) = 1 - 2^{-p(n)}$ 。

6.7 (ZPP 类的另一种定义) 称判定问题 S 可在期望的概率多项式时间内求解, 如果存在一个随机算法 A 及多项式 p , 使得对任意 $x \in \{0, 1\}^*$, 有 $\Pr[A(x) = \chi_S(x)] = 1$ 且 $A(x)$ 的期望运行步骤数至多为 $p(|x|)$ 。证明 $S \in ZPP$ 当且仅当 S 可在期望的概率多项式时间内求解。

提示: 重复调用 ZPP-算法, 直至其产生一个不同于 $\perp$ 的输出, 从而得到期望的概率多项式时间算法。另一方面, 一旦概率多项式时间算法的运行步数超过了期望运行步数的 2 倍, 则将其截断(并输出 $\perp$), 从而得到一个 ZPP-算法。

6.8 证明对任意 $S \in \mathcal{NP}$, 存在一个概率多项式时间算法 A , 使得对任意 $x \in S$ 有 $\Pr[A(x)=1] > 0$, 而对任意 $x \notin S$, 有 $\Pr[A(x)=0]=1$ 。也就是说, A 对于“是”实例的错误概率至多为 $1 - \exp(-\text{poly}(|x|))$, 而对“否”实例从不出错。因此, 可以虚拟地认为 $\mathcal{NP}$ 类有极大的单边错误概率。

6.9 令 BPP 和 $\text{co}RP$ 为承诺问题类(如定理 6.9)。

(1) 证明 BPP 中每个问题可以归约到集合 $\{1\} \in \mathcal{P}$, 这里的归约是双边错误的随机 Karp-归约。

(2) 证明如果一个集合 S 可以确定地 Karp-归约到 RP (或 $\text{co}RP$), 则 $S \in RP$ (或 $S \in \text{co}RP$)。

6.10 (具有高效随机性的错误缩减) 注意标准的错误缩减(如习题 6.4)得到的错误概率为 δ 时, 需要向算法的随机性复杂度中增加一个因子 $O(\log(1/\delta))$ 。利用附录 D.4.1.3 中具有高效随机性的错误缩减, 证明为得到错误概率 δ , 只需将算法的随机性复杂度由 r 增加至 $O(r) + 1.5 \log_2(1/\delta)$ 即可。注意: 这考虑到了满足定理 6.9 证明中说明性段落里的假设。

6.11 续定理 6.9 证明中的说明性段落, 考虑承诺问题 $\Pi' = (\Pi'_{\text{yes}}, \Pi'_{\text{no}})$, 满足 $\Pi'_{\text{yes}} = \{(x, r') : |r'| = p'(|x|) \wedge (\forall r'' \in \{0, 1\}^{|r'|}) A'(x, r' r'') = 1\}$, 而 $\Pi'_{\text{no}} = \{(x, r') : x \notin S\}$ 。回顾对每个 x , 有 $\Pr_{r \in \{0, 1\}^{2p'(|x|)}} [A'(x, r) \neq \chi_S(x)] < 2^{-(p'(|x|)+1)}$ 。

(1) 证明 x 到 (x, r') 的映射构成一个 S 到 Π' 的单边错误随机 Karp-归约, 其中 r' 在 $\{0, 1\}^{p'(|x|)}$ 上均匀分布。

(2) 证明 Π' 属于承诺问题类 coRP 。

6.12 ($\mathcal{NP}$ 的随机版本), 考虑以下对 $\mathcal{MA}$ 的两种变形 (其中考虑到了 $\mathcal{NP}$ 的主要随机化版本)。

(1) $S \in \mathcal{MA}^{(1)}$, 如果存在一个概率多项式时间算法 V , 使得对任意 $x \in S$, 存在 $y \in \{0, 1\}^{\text{poly}(|x|)}$, 使得 $\Pr[V(x, y) = 1] \geq 1/2$, 而对任意 $x \notin S$ 及所有 y , 有 $\Pr[V(x, y) = 0] = 1$ 。

(2) $S \in \mathcal{MA}^{(2)}$, 如果存在一个概率多项式时间算法 V , 使得对任意 $x \in S$, 存在 $y \in \{0, 1\}^{\text{poly}(|x|)}$, 使得 $\Pr[V(x, y) = 1] \geq 2/3$, 而对任意 $x \notin S$ 及所有 y , 有 $\Pr[V(x, y) = 0] \geq 2/3$ 。

证明: $\mathcal{MA}^{(1)} = \mathcal{NP}$, 而 $\mathcal{MA}^{(2)} = \mathcal{MA}$ 。

提示: 对第一部分, 注意使 V 接受 (x, y) 的内部随机数序列可以合并到 y 中 (得到一个标准的 NP-证据)。对第二部分, 应用定理 6.9 的证明思想, 并注意到一个充分的变换序列 (验证者使用的) 可以与由证明者发出的单个消息合并起来。

6.13 ($\mathcal{BPP} \subseteq \mathcal{ZPT}^{\mathcal{NP}}$) 续定理 6.9 的证明, 构造一个由 $\mathcal{BPP}$ 到 $\mathcal{NP}$ 的零边错误随机归约, 其中所有的类都是标准的判定问题类。

提示: 对于输入 x , ZPP-算法均匀选择 $\bar{s} = (s_1, s_2, \dots, s_m)$, 并对每个 $\sigma \in \{0, 1\}$, 发出询问 $(x, \sigma, \bar{s})$, 如果对任意 r , 有 $\bigvee_i (A(x, r \oplus s_i) = \sigma)$, 则 (coNP) 预言会对询问做出肯定回答。算法输出 σ 当且仅当询问 $(x, \sigma, \bar{s})$ 的应答是肯定的, 否则, 输出 $\perp$ (即对两个询问的回答都是否定的)。

6.14 ($\mathcal{BPP}$ 的承诺版本的完全性) 对于 $\mathcal{BPP}$ 的承诺版本, 构造一个承诺问题, 使其在 (确定的对数空间) Karp-归约下是完全的。

提示: 包含 “是” 实例的承诺问题是以至少 $2/3$ 的概率接受所有输入的布尔电路, 而包含 “否” 实例的承诺问题是以至少 $2/3$ 的概率拒绝所有输入的布尔电路。归约本质上就是定理 2.21 证明中所构造的归约, 而承诺用于构造一个 BPP-算法。

6.15 (BP_{TIME} 的承诺问题版本的层级定理) 给定一种计算模型, 令 $\text{BP}_{\text{TIME}}(t)$ 表示可由时间复杂度为 t 的随机算法求解的承诺问题类, 且算法的双边错误概率不超过 $1/3$ (标准定义只针对判定问题)。阐述并证明类似于定理 4.3 和引理 4.4 的结论。

提示 (由 Dieter van Melkebeek 给出): 应用定理 4.6 证明中使用的 “延迟对角线” 方法, 而非定理 4.3 证明中的简单对角线方法。类似于定理 4.6 的证明, 对每个 $\sigma \in \{0, 1\}$, 如果 $\Pr[M'(x) = \sigma] \geq 2/3$, 则定义 $A_M(x) = \sigma$, 否则, 定义 $A_M(x) = \perp$ (即当 $1/3 < \Pr[M'(x) = 1] < 2/3$ 时), 其中 $M'(x)$ 表示截取 $M(x)$ 的前 $t_1(|x|)$ 步所得到的计算。对 $x \in [\alpha_M, \beta_M - 1]$, 定义 $f(x) = A_M(x+1)$, 其中 $f(x) = \perp$ 的含义是 x 违背了承诺。如果

$A_M(\alpha_M)=0$, 则定义 $f(\beta_M)=1$, 否则, 定义 $f(\beta_M)=0$ (即如果 $A_M(\alpha_M) \in \{1, \perp\}$)。注意: 如果当 $x \in [\alpha_M, \beta_M - 1]$ 时模仿 $M'(x)$ 的单步计算, 而当 $x = \beta_M$ 时模仿 $M'(\alpha_M)$ 的全部计算, 则可在随机时间 $\tilde{O}(t_1(|x|+1))$ 内计算 $f(x)$ 。证明承诺问题 f 不可能在随机时间 t_1 内计算, 注意到 β_M 满足承诺, 而对每个满足承诺的 $x \in [\alpha_M + 1, \beta_M]$ (即 $f(x) \in \{0, 1\}$), 如果 $A_M(x) = f(x)$, 则 $f(x-1) = A_M(x) \in \{0, 1\}$ 。

6.16 (提取模素数的平方根) 利用如下提示构造一个概率多项式时间算法, 对于输入的素数 P 和模 P 的平方剩余 s , 算法返回满足 $r^2 \equiv s \pmod{P}$ 的 r 。

(1) 证明如果 $P \equiv 3 \pmod{4}$, 则 $s^{(P+1)/4} \pmod{P}$ 是平方剩余 $s \pmod{P}$ 的一个平方根。

(2) 注意: 第一项取决于能找到一个奇数 e , 满足 $s^e \equiv 1 \pmod{P}$ 。事实上, 如果能找到这样的 e , 则可以输出 $s^{(e+1)/2} \pmod{P}$ (在第一项中, 使用 $e = (P-1)/2$, 这是一个奇数, 因为 $P \equiv 3 \pmod{4}$)。

证明只需找到一个奇整数 e , 剩余 t 以及一个偶数 e' 满足 $s^e t^{e'} \equiv 1 \pmod{P}$ 即可, 因为 $s \equiv s^{e+1} t^{e'} \equiv \left(s^{(e+1)/2} t^{e'/2}\right)^2$ 。

(3) 给定素数 $P \equiv 1 \pmod{4}$, 平方剩余 s , 以及任意一个非平方剩余 t (即 $t^{(P-1)/2} \equiv -1 \pmod{P}$), 证明第 2 项中的 e 和 e' 可以高效地找到。^①

(4) 证明对于素数 P , 一个均匀选择的 $t \in \{1, 2, \dots, P\}$ 满足 $t^{(P-1)/2} \equiv -1 \pmod{P}$ 的概率是 $1/2$ 。

注意: 只在最后一项中使用了随机化, 而且只对 $P \equiv 1 \pmod{4}$ 的情形使用。

6.17 针对算术电路 (见附录 B.3) 的定义, 证明以下判定问题属于 coRP : 给定一对深度为 d 的电路 (C_1, C_2) , 电路的基域中元素个数多于 2^{d+1} , 判定这两个电路是否能计算同一个多项式。

提示: 注意: 由这些电路计算的多项式次数至多为 2^d 。

6.18 (小空间随机计步器) 在习题 4.7 中, 定义计步器为一个算法, 在发出一定数量的“符号”之后停止, 这些符号被定义为进入 (并离开) 一个指定的 (算法) 状态。计步器可能与其他程序并行执行, 从而 (其他程序) 在运行了预先确定的若干步之后可以挂起。注意: 存在一个简单的确定机器, 对于输入 n , 在发出 n 个符号之后停机, 其使用的空间为 $O(1) + \log_2 n$ (使用的时间为 $\tilde{O}(n)$)。本题的目标是要构造一个 (随机的) 计步器, 在使用同样的空间时, 可以发出更多的符号。具体地, 构造一个 (随机) 算法, 对于输入 n , 使用 $O(1) + \log_2 n$ 的空间 (以及 $\tilde{O}(n)$ 的时间), 在发出期望的 2^n 个符号之后停止。进一步地, 证明该计步器在发出的符号数量位于 2^{n-k} 和 2^{n+k} 之间时, 以至少 $1 - 2^{-k+1}$ 的概率停止。

提示: 重复上述实验直至成功。每次实验需要均匀选择 n 比特 (即随机掷 n 次硬币), 如果所有比特都为 1 (即每次掷硬币结果都为正面), 则认为实验成功。注意: 这个实验可在空间 $O(1) + \log_2 n$ 内实现 (主要用于实现判定比特数量的标准计数器)。因此, 每次实验成功的

① 记 $(P-1)/2 = (2j_0+1) \cdot 2^{i_0}$, 并注意到 $s^{(2j_0+1)2^{i_0}} \equiv 1 \pmod{P}$, 这个条件可以写成 $s^{(2j_0+1)2^{i_0}} t^{(2j_0+1)2^{i_0-1}} \equiv 1 \pmod{P}$ 。给定某个 $i' > i_0$ 及 j' , 有 $s^{(2j_0+1)2^{i'}} t^{(2j'+1)2^{i'}}$ (说明如何找到 $i'' > i-1$ 和 j'' , 使得 $s^{(2j_0+1)2^{i''-1}} t^{(2j'+1)2^{i''}} \equiv 1 \pmod{P}$) (提示: $s^{(2j_0+1)2^{i-1}} t^{(2j'+1)2^{i-1}} \equiv \pm 1 \pmod{P}$ 以及 $t^{(2j_0+1)2^{i_0}} \equiv -1 \pmod{P}$)。应用这个推理 i_0 次, 便得到了所需结论。

概率为 2^{-n} , 期望的实验次数为 2^n 。

6.19 (对任意无向图上随机漫游的分析) 为了完成命题 6.13 的证明, 证明如果 $\{u, v\}$ 是图 $G=(V, E)$ 的一条边, 则 $E[X_{u,v}] \leq 2|E|$ 。回顾对于一个固定的图, $X_{u,v}$ 是一个随机变量, 表示从结点 u 开始的随机漫游在第一次遇到 v 时经过的边数。

提示: 令 $Z_{u,v}(n)$ 为随机变量, 对长度为 n 的随机漫游中出现 u 与 v 时, 其间的最短路径数量计数。其中漫游开始于固定的结点 (对于非二部图良好定义, 这个条件又可通过增加一个自环来强制满足)。一方面, $E[X_{u,v} + X_{v,u}] = \lim_{n \rightarrow \infty} (n/E[Z_{u,v}(n)])$, 这是由于漫游的无记忆性; 另一方面, 如果在漫游的第 i 步经过边 $\{u, v\}$, 方向由 v 到 u , 则令 $\chi_{v,u}(i) \stackrel{\text{def}}{=} 1$, 否则, 令 $\chi_{v,u}(i) \stackrel{\text{def}}{=} 0$, 则有 $\sum_{i=1}^n \chi_{v,u}(i) \leq Z_{u,v}(n) + 1$ 且 $E[\chi_{v,u}(i)] = 1/2|E|$ (因为在每一步, 每条有向边出现于漫游中的概率相同), 从而 $E[X_{u,v}] < 2|E|$ 。

6.20 (类 $PP \supseteq BPP$ 及其与 $\#P$ 的关系) BPP 类涉及到有用的概率多项式时间算法, 与之相反, PP 类与此类算法无关, 它更接近于 $\#P$ 。判定问题 S 属于 PP , 如果存在一个概率多项式时间算法 A , 使得对每个 $x, x \in S$ 当且仅当 $\Pr[A(x)=1] > 1/2$ 。注意: $BPP \subseteq PP$ 。证明 PP 可以 Cook-归约为 $\#P$, 反之亦然。

提示: 对 $S \in PP$ (利用算法 A), 考虑关系 $R, (x, r) \in R$ 当且仅当 A 的随机输入为 $r \in \{0, 1\}^{p(|x|)}$ 时接受输入 x , 其中 p 为一个适当的多项式。因此, $x \in S$ 当且仅当 $|R(x)| > 2^{p(|x|)-1}$, 这又可由向 R 的计数函数发出询问来确定。为了将 $f \in \#P$ 归约到 PP , 考虑由 f 计数的关系 $R \in PC$ (即 $f(x) = |R(x)|$), 以及命题 6.15 中定义的判定问题 S_f 。令 p 为定义 R 的解长度的多项式 (即 $(x, y) \in R$ 蕴含着 $|y| = p(|x|)$), 考虑如下算法 A' : 对于输入 (x, N) , A' 以 $1/2$ 的概率均匀选择 $y \in \{0, 1\}^{p(|x|)}$, 当且仅当 $(x, y) \in R$ 时接受, 否则, 算法 A' (以剩余的 $1/2$ 概率) 接受的概率恰为 $\frac{2^{p(|x|)} - N + 0.5}{2^{p(|x|)}}$ 。证明 $(x, N) \in S_f$ 当且仅当 $\Pr[A'(x)=1] > 1/2$ 。

6.21 (枚举问题) 对任意二元关系 R , 定义 R 的枚举问题为函数 $f_R: \{0, 1\}^* \times \mathbb{N} \rightarrow \{0, 1\}^* \cup \{\perp\}$, 使得当 $|R(x)| \geq i$ 时, $f_R(x, i)$ 等于 $|R(x)|$ 中第 i 个元素, 否则, $f_R(x, i) = \perp$ 。上述定义针对着串的标准词典序, 但也可以使用其他有效的排序方式。^①

(1) 证明对任意多项式界限 R , 计算 $\#R$ 可以归约为计算 f_R 。

(2) 证明对任意 $R \in PC$, 计算 f_R 可以归约为 $\#P$ 中某个问题。

提示: 考虑二元关系 $R' = \{(\langle x, b \rangle, y) : (x, y) \in R \wedge y \leq b\}$, 并证明 f_R 可以归约为 $\#R'$ 。

(特别提示: 注意到 $f_R(x, i) = y$ 当且仅当 $|R'(\langle x, y \rangle)| = i$ 且对每个 $y' < y$, 有 $|R'(\langle x, y' \rangle)| < i$ 。)

6.22 (虚构的 $\#P$ -完全问题) 证明存在一个关系 $R \in PC$, 使得 $\#R$ 是 $\#P$ -完全的, 且

^① 串的序是一种从自然数集到所有串集的双射 μ 。称一种序为有效的, 如果 μ 及其逆都可以高效计算。标准词典序满足 $\mu(i) = y$, 如果串 $1y$ 是整数 i 的 (紧凑) 二元扩张, 即 $\mu(1) = \lambda$, $\mu(2) = 0$, $\mu(3) = 1$, $\mu(4) = 00$ 等。

$S_R = \{0, 1\}^*$ 。进一步地, 证明对任意 $R' \in PC$, 存在 $R \in PF \cap PC$, 使得对任意 x , 有 $\#R(x) = \#R'(x) + 1$ 。注意: 如果从任意使 $\#R'$ 为 $\#P$ -完全的关系 $R' \in PC$ 开始, 则得到了定理 6.19。

6.23 (整矩阵积和式的计算) 证明求 0-1 矩阵的积和式在计算上等价于求二部图的完美匹配数量。

提示: 给定一个二部图 $G = ((X, Y), E)$, 考虑表示 X 与 Y 之间边的矩阵 M (即如果 X 的第 i 个结点与 Y 的第 j 个结点是连通的, 则 M 中的第 (i, j) 项为 1), 并注意到只有 G 的完美匹配才对 M 的积和式计算有影响。

6.24 (求模 3 的积和式) 将命题 6.21 与定理 6.29 相结合, 证明对任意固定的 $n > 1$, 且 n 不能被 c 的任何幂整除, 在随机归约下, 计算模 n 的积和式是 NP-困难的。由于命题 6.21 对 $c = 2^{10}$ 成立, 所以对任意非 2 的幂的整数 $n > 1$, 困难性都成立 (我们指出, 另一方面, 对任意固定的 $n = 2^e$, 模 n 的积和式可在多项式时间内计算 (见参考文献[230]中定理 3))。

提示: 将命题 6.21 中的归约用于判定是否一个 3CNF 范式具有唯一可满足赋值或不可满足的承诺问题。注意: 对任意 m , 有 $c^m \not\equiv 0 \pmod{n}$

6.25 (命题 6.21 中的负值) 设 $P \neq NP$, 证明命题 6.21 对于只包含非负整数的集合 I 不成立。注意即使当集合 I 为无限集时 (甚至当 I 是所有非负整数构成的集合时), 这个断言也成立。

提示: 命题 6.21 中的归约构成了 3SAT 到判定 I 上矩阵的积和式是否非 0 的 Karp-归约。注意: 非负矩阵的积和式非零当且仅当相应的二部图有完美匹配。

6.26 (对于积和式归约的高级分析) 证明命题 6.21 证明中提出的高级归约的正确性。即证明如果子句满足证明中的 3 个条件, 则 ϕ 的每个可满足的赋值对于 G_ϕ 的 SWCC 恰好贡献 c^m , 而不可满足的赋值无贡献。

提示: 根据所使用的漫游轨迹中的边将 G_ϕ 中的圈覆盖聚集起来 (即属于各种轨迹的圈覆盖中边)。注意: 这些边与外部边之间的一致性/相关性。利用假设条件 (关于子句分图), 证明对每个这样的轨迹边集 T , 如果所有圈覆盖的权重之和非零, 则以下结论成立。

(1) T 与出现在每个特定子句分图中的轨迹边集的交集非空。进一步地, 如果该集合中包含某个进入结点 (或外出结点) 的一条进入边 (或外出边), 则其中必然也包含相应外出结点 (或进入结点) 的外出边 (或进入边)。

(2) 如果 T 中包含一条属于某个轨迹的边, 则其中包含该轨迹中的所有边, 从而对每个变量 x , 集合 T 中包含与 x 相关的单个轨迹中的边。

(3) 由 T “选中” 的轨迹对应于 ϕ 中变量的单个真值赋值, 且这种赋值满足 ϕ (因为对每个子句, T 中包含一条外部边, 对应于满足该子句的文字)。

注意: 上述类型的不同集合可以得到不同的可满足赋值, 且每种可满足赋值也由某个上述类型的集合得出。

6.27 (对子句分图实现的分析) 证明在命题 6.21 证明中提出的子句实现的正确性。即证明如果盒子满足证明中的 3 个假设条件, 则图 6.4 中的子句也满足相应条件。

提示: 将相应于使用不属于盒中边集的分图中的圈覆盖聚集起来, 不属于盒的边如图 6.4 所示。利用 (关于盒子的) 假设条件证明, 对于每个由不属于盒的边构成的集合 S , 如果所有使用不属于盒的边集 S 的圈覆盖的权重之和非零, 则有以下结论成立。

(1) S 与每个盒中边集的交集一定包含两条 (非自环的) 边, 每条边依附于盒子的一个

顶点。当然，一条边为进入边，另一条为外出边。将（分图中）6个指定结点与盒子端点相连的六条边称为连接边，注意：如果 S 中包含一个依附于某个盒子端点的连接边，则其中必包含依附于另一个端点的连接边，此时称该盒子被 S 所“选中”。

（2）未被 S 选中的3个（由文字指定的）盒子中，每一个都被由左向右“横穿”（即圈覆盖中包含左端点的一条进入边和右端点的一条外出边）。因此，集合 S 中必包含一个连接边，因为否则将不存在任何有向圈能覆盖图6.4中最左边的结点，从而， S 必选中某个盒子。

（3）集合 S 完全由其选中盒子构成的非空集所确定。

当 $c=b^5$ 时，子句分图的假设性质得证。

6.28 （对子句分图指定盒子的分析）证明式（6.4）中的 4×4 矩阵满足命题6.21证明中第二部分使用的关于“盒子”的假设性质。特别地：

（1）证明在盒子条件与图的关联矩阵某个子矩阵的积和式值所满足条件间存在对应关系。

提示：例如，证明第一个条件对应于要求整个矩阵的积和式值为0，第二个条件对应于去掉第一行和第四列或去掉第四行和第一列所得到的子阵。

（2）验证式（6.4）中的矩阵满足上述条件（关于某个子阵的积和式值）。

证明不存在满足上述条件的 3×3 矩阵（以及 2×2 矩阵）。

6.29 （近似计数的错误缩减）证明定义6.24中的错误概率 δ 可由 $1/3$ （或甚至是 $(1/2) + (1/\text{poly}(|x|))$ ）减小为 $\exp(-\text{poly}(|x|))$ 。

提示：调用较弱算法足够多次，并对调用中得到的值求平均值。

6.30 （近似与缝隙问题）令 $f: \{0,1\}^* \rightarrow \mathbb{N}$ 为多项式界函数（即 $|f(x)| = \text{poly}(|x|)$ ）， $\varepsilon: \mathbb{N} \rightarrow [0,1]$ 为多项式时间计算的不可忽略函数。证明与关系 $R_{f,\varepsilon} \stackrel{\text{def}}{=} \{(x,v): |v - f(x)| \leq \varepsilon(|x|) \cdot f(x)\}$ 相关的搜索问题在计算上等价于区分集合 $\{(x,v): v < (1 - \varepsilon(|x|)) \cdot f(x)\}$ 与集合 $\{(x,v): v > (1 + \varepsilon(|x|)) \cdot f(x)\}$ 中元素的（“缝隙”）承诺问题。

6.31 （对某些#P-完全问题的强近似）证明存在一些#P-完全问题（虽然是一些非自然的问题），对其 $(\varepsilon, 0)$ -近似可由（确定的）多项式时间算法找到。进一步地，算法的运行时间是 $1/\varepsilon$ 的多项式。

提示：将任一关于某个 $R_1 \in \text{PC}$ 的#P-完全问题与一个平凡的计数问题（如与平凡关系 $R_2 = \bigcup_{n \in \mathbb{N}} \{(x,y): x,y \in \{0,1\}^n\}$ 相关的计数问题）相结合。不失一般性，证明 $\#R_1(x) \leq 2^{|x|/2}$ 。证明 $R = \{(x,1y): (x,y) \in R_1\} \cup \{(x,0y): (x,y) \in R_2\}$ 的计数问题是#P-完全的（通过由 $\#R_1$ 的归约）。构造一个确定算法，对于输入的 x 及 $\varepsilon > 0$ ，在时间 $\text{poly}(|x|/\varepsilon)$ 内输出关于 $\#R(x)$ 的 $(\varepsilon, 0)$ -近似。

（附加提示：区分 $\varepsilon \geq 2^{-|x|/2}$ 与 $\varepsilon < 2^{-|x|/2}$ 的情形）。

6.32 （DNF可满足性的相关近似）参考6.2.2.1节，证明以下断言。

（1）两个假设都考虑到了 $S_i = C_i^{-1}(1)$ 时的一般环境，其中 $C_i^{-1}(1)$ 表示满足合取式 C_i 的真值赋值集合。

提示：在证明第二个假设时，注意到其中将以下两个假设归约为合取式。

① 给定 i ，可以高效地生成均匀分布的 S_i 中元素。实际上，生成 S_i 上的均匀分布就足

够了。

② 给定 i 和 x , 可以高效地确定是否有 $x \in S_i$ 。

(2) 证明命题 6.26, 其中涉及到一些细节, 如构造 6.25 实现时的错误概率。

(3) 注意: 构造 6.25 不要求对 $|S_i|$ 的精确计数。当只能对 $|S_i|$ 进行因子为 $1 \pm \varepsilon'$ 的近似计数时, 分析其输出分布。

6.33 (在近似计数中减小相关偏移) 证明对任意 $R \in PC$, 任意多项式 p 及常数 $\delta < 0.5$, 存在 $R' \in PC$, 满足对 $\#R$ 的 $(1/p, \delta)$ -近似计数可归约为对 $\#R'$ 的 $(1/2, \delta)$ -近似计数。进一步地, 对任意 $F(n) = \exp(\text{poly}(n))$, 证明存在 $R'' \in PC$, 使得对 $\#R$ 的 $(1/p, \delta)$ -近似计数可归约为对 $\#R'$ 的近似计数, 且近似因子为 F , 错误概率为 δ 。

提示 (主要部分): 对于 $t(n) = \Theta(p(n))$, 定义 R' 使得 $(y_1, y_2, \dots, y_{t(|x|)}) \in R'(x)$ 当且仅当 $(\forall i) y_i \in R(x)$ 。注意: $|R(x)| = |R'(x)|^{1/t(|x|)}$, 从而如果 $a = (1 \pm (1/2)) \cdot |R'(x)|$, 则 $a^{1/t(|x|)} = (1 \pm (1/2))^{1/t(|x|)} \cdot |R(x)|$ 。

6.34 (近似计数的偏移归约) 续习题 6.33, 证明如果可通过简化的归约证明 R 的 NP-完全性, 则对任意正多项式 p 及常数 $\delta < 0.5$, $\#R$ 的 $(1/p, \delta)$ -近似问题可以归约为 $\#R$ 的 $(1/2, \delta)$ -近似问题。

提示: 将习题 6.33 中的归约 (到 $\#R'$ 的 $(1/2, \delta)$ -近似) 与 $\#R'$ 到 $\#R$ 的简化归约相结合。

证明对任意满足 $F'(n) = \exp(n^{o(1)})$ 函数 F' , 我们也可将上述问题归约为 $\#R$ 的近似问题, 使其与 F' 相差一个因子, 且错误概率为 δ 。

提示: 利用如习题 6.33 中的 R'' , 会遇到一个技术难点。将从 $\#R$ 到 $\#R'$ 的 (“放大的”) 归约与 $\#R''$ 到 $\#R$ 的简化归约相结合, 会增加实例的长度。事实上, 新实例的长度是原始实例长度的多项式, 但是这个多项式可能取决于 R'' , 而这又依赖于 F' 。因此, 我们不能使用 $F'(n) = \exp(n^{1/O(1)})$, 但可以用 $F'(n) = \exp(n^{o(1)})$ 。

6.35 针对定理 6.27 证明中的过程, 说明如何利用一个 NP-预言来判断是否 “通过一个随机筛” 的解的数量大于 t 。可以使用长为 x, h 和 t 长度多项式的询问。

提示: 考虑集合 $S'_{R,H} \stackrel{\text{def}}{=} \{(x, i, h, l') : \exists y_1, y_2, \dots, y_l \text{ 使得 } \psi'(x, h, y_1, y_2, \dots, y_l)\}$, 其中 $\psi'(x, h, y_1, y_2, \dots, y_l)$ 成立当且仅当各个 y_j 是不同的, 且对任意 j , 有 $(x, y_j) \in r \wedge h(y_j) = 0^j$ 。

6.36 (简化归约与定理 6.29) 通过证明存在搜索问题 $R \in PC$, 使得 PC 中每个问题都可归约到 R (通过非简化归约) 且承诺问题 $(US_R, \bar{S}_R)$ 仍可在多项式时间内判定来说明定理 6.29 证明中简化归约的重要性。

提示: 考虑如下关于 SAT 的虚构证据关系, 其中当 $\sigma \in \{0, 1\}$ 且 τ 满足 ϕ 时 $(\phi, \sigma\tau) \in R$ 。注意: SAT 的标准证据关系可以归约到 R , 但是这种归约是非简化的。还要注意 $US_R = \phi$, 从而显然 $(US_R, \bar{S}_R)$ 。

6.37 续习题 6.36, 证明存在一个搜索问题 $R \in PC$, 使得 $\#R$ 为 $\#P$ -完全的, 而承诺问题 $(US_R, \bar{S}_R)$ 仍可在多项式时间内判定。对于 R 为 PC -完全的情形, 提出一种证明, 而当 $R \in PF$

时, 提出另一种证明。

提示: 对第一种情况, 习题 6.36 提示中的关系 R 可以满足要求。对第二种情形, 根据定理 6.20, 并考虑当 R 为相应的完美匹配关系时, 判定 $(US_R, \bar{S}_R)$ 是容易的 (通过计算行列式) 这一事实。

6.38 证明 SAT 可以随机归约为对 SAT 唯一解的判定, 这里不能利用 SAT 在简化归约下为 NP-完全问题这一事实。

提示: 根据定理 6.29 的证明, 并使用构造 D.3 中提出的两两独立的 Hash 函数类。注意: 在这种情况下, 条件 $(\tau \in R_{SAT}(\phi)) \wedge (h(\tau) = 0^i)$ 可以直接被编码为一个 CNF 范式。也就是说, 考虑公式 ϕ_h , 满足 $\phi_h(z) \stackrel{\text{def}}{=} \phi(z) \wedge (h(z) = 0^i)$; 并注意 $h(z) = 0^i$ 可以写成 i 个条件的合取, 其中每个条件是一个 CNF, 在逻辑上等价于 z 中某些比特的奇偶校验 (而这些比特的一致性可由 h 确定)。

6.39 (只有少数解的搜索问题) 对任意 $R \in PC$ 及任意多项式 p , 考虑承诺问题 $(\text{Few}S_{R,p}, \bar{S}_R)$, 其中 $\text{Few}S_{R,p} \stackrel{\text{def}}{=} \{x : |R(x)| \leq p(|x|)\}$ 。

(1) 证明对任意 $R \in PC$, 定理 6.27 证明中提出的 (近似计数) 程序蕴含了一个 S_R 到 $(\text{Few}S_{R',100}, \bar{S}_{R'})$ 的随机归约, 其中 $R'(x, i, h) = \{y \in R(x) : h(x) = 0^i\}$ 。

(2) 对任意 $R' \in PC$ 及任意多项式 p' , 构造一个从 $(\text{Few}S_{R',p'}, \bar{S}_{R'})$ 到 $(US_{R'}, \bar{S}_{R'})$ 的随机归约, 其中 $R''(x', m) = \{y_1 < y_2 < \dots < y_m : (\forall j) y_j \in R'(x')\}$ 。

提示: 将 x' 映射到 (x', m) , 其中 m 在 $[p'(|x'|)]$ 中均匀选择。

6.40 证明与完美匹配相关的搜索问题满足定理 6.31 中的假设。

提示: 对于给定的图 $G = (V, E)$, 将每个匹配编码为一个 $|E|$ -比特的串, 其中令第 i 个比特为 1 当且仅当第 i 条边在匹配中。

6.41 (另一个近似计数程序) 应用构造 6.32 中的第一步得到 $\#R$ 的一个近似计数程序。

提示: 对于 $m = 0, 1, \dots, l$, 调用构造 6.32 的第一步直至得到一个否定回答, 并对当前的 m , 输出 $120l \cdot 2^m$ 。当 $|R(x)| > 80l$ 时, 这样做可以得到 $|R(x)|$ 的常数因子近似。事实上, 在第 m 次迭代中, 通过发出更多的询问, 可以得到一个更好的近似 (即对 $i = 10l, \dots, 120l$, 发出形如 $(x, h, 1^i)$ 的询问)。对于 $|R(x)| \leq 80l$ 的情形, 可以利用构造 6.32 的第二步, 此时将得到一种精确计数。

6.42 设 R 为任意一个 PC -完全搜索问题, 证明 R 的近似计数与均匀生成可以随机归约为 S_R 的成员判定, 其中近似计数的含义是 $(1 - (1/p))$ -近似, p 为任意多项式。

提示: 注意: 由构造 6.32 可以得到这样的程序 (还可见于习题 6.41), 除了向 NP 中其他一些集合发出询问之外。利用 S_R 的 NP-完全性可证。

6.43 构造一个概率多项式时间算法, 能解 R_{dnf} 的均匀生成问题, 其中当 ϕ 为 DNF 范式且 τ 为其可满足性赋值时, $(\phi, \tau) \in R_{dnf}$ 。

提示: 考虑集合 $\{(i, e) : i \in [m] \wedge e \in S_i\}$, 其中 S_i 如 6.2.2.1 节中所定义。

第 7 章

困难性的用途

她这样说完，把精美的绳鞋系到脚上，那是双奇妙的金鞋，能使女神飞越大海和无垠的陆地，像疾风一样轻快。她然后又抓起巨矛，铆有锐利的铜尖粗长、硕大、沉重，她用它扫荡英雄们的战斗行列，因为他们，使主神的这位目光炯炯的女儿发怒。

荷马，奥德赛，1:96–101

自然界中存在着不可能（或看上去不可能）求解的计算问题，这意味着有些事情是人们做不到的。但其中也有积极的一面，因为人们可以根据需要，让困难问题“发挥作用”，困难问题最著名的应用是在密码学领域。

在应用困难问题时要求能高效地生成困难实例，而最坏情况下的困难性并不能保证这一点。换言之，这里涉及到“偶然”困难性（即最坏情况或平均情况的困难性）与“典型”困难性（关于某些易处理分布）之间的缝隙。本章将用大部分篇幅讨论如何桥接这个缝隙，这就是“困难放大”。困难性的典型应用可见于第 8 章及附录 C 中。

内容提要

我们考虑两个与 $P \neq NP$ 有关的猜想。第一个猜想是存在一些指数时间可解的问题（即属于 ϵ ），不能用较小（如多项式规模）的（非一致）电路类求解。我们将说明，这种最坏情况下的猜想可以转化为平均情况下的困难性结论；具体地，可以得到一些谓词，利用较小电路是强“不可近似”的。利用这些谓词，可以对 BPP 类非平凡地去随机化（见 8.3 节）。

第二个猜想是 NP 中存在问题（即 PC 中的搜索问题），易于生成（已解的）实例，但（对没有生成这些实例的参与方而言）很难求解。这一猜想可用单向函数的概念来理解，后者是指易求值但（在平均意义上）难求逆的函数。我们证明，从一个在温和平均意义上难求逆的函数出发，可以构造一个在较强平均意义上难求逆的函数，而由后者可以得到非常难以求近似的谓词（称为困难核心谓词）。这些谓词可用于构造通用的伪随机数发生器（见 8.2 节），以及许多其他的密码学应用中（见附录 C）。

本章在具体阐述时将掉换上述两个猜想的顺序：先讨论单向函数（7.1 节），然后介绍 ϵ 中用小规模电路难求解的问题（7.2 节）

教学注释：采用上述顺序的原因如下。首先，单向函数在概念上很吸引人，并且实用价值较高。其次，单向函数中的困难放大比 ε 中的困难放大在技术上更简单（事实上，7.2 节是本书技巧性最强的部分）。再次，讨论两种猜想时共同使用的某些技术在单向函数的背景下似乎更易于理解。最后，采用这种顺序使得困难放大成为一个可选的教学内容，而我们建议把单向函数作为必选教学内容（出于上述理由）。

如果希望在教学中选讲困难放大及伪随机性，我们建议采用当前顺序。即先讲述困难放大，再讲述伪随机性。

预备知识

读者要熟悉概率论基础（见附录 D.1）及随机算法（见 6.1 节）。特别地，本章内容涉及到随机变量（附录 D.1.1）及各种“大数定律”（附录 D.1.2）。

7.1 单向函数

不严格地讲，单向函数是一些易于求值但（在平均情况下）难求逆的函数。因此，如果假设单向函数存在，则可以认为存在难求逆的高效算法（即函数的正向计算）。日常生活中有大量类似现象（如火柴的燃烧）。因此，单向函数存在性的假设是日常经验在复杂性理论中的类比。

也可以认为单向函数是生成不可能解决的“难题”的有效方法，即这种难题是单向函数的一个随机映像，而其解为相应的原像。另外，生成难题的人知道其解，并能高效地验证（其他可能的）解的正确性。事实上，如 7.1.1 节所述，生成这种难题的每种途径都可以转化为一个单向函数。

读者要注意，使用生成难题的术语阐述单向函数，这明显属于密码学风格。事实上，单向函数是密码学的核心，但是本章对此不做深入讨论（可以参考附录 C）。简单说来，单向函数与（一般意义上的）伪随机数发生器密切相关，8.2 节将研究这种联系。本节我们主要讨论单向函数本身。

教学注释：我们建议在复杂性理论课程中介绍伪随机性的基本内容，但不建议也介绍密码学。原因在于密码学远比伪随机性复杂得多（可以比较安全加密与伪随机数发生器的定义）。这种复杂性源于密码学具有极其丰富的概念，这当然是有益的，然而，密码学中某些至关重要的概念在复杂性理论中却并不重要，因此，在复杂性理论课程中讲授密码学或许会为本课程增加额外的负担，还可能使读者对密码学产生浮浅甚至是错误的理解。我们不太确定这两种可能性哪种更糟糕，然而，仍将密码学基础知识放在附录中（见附录 C），供感兴趣的读者参考。

7.1.1 困难实例的生成与单向函数

本章在一开始预言困难问题可以发挥作用，在此检验这个预言。一个基本思路是困难问题可以为敌对方的计算提供障碍，从而保护自己一方的利益。这里的敌对方可以是实际存在

的（如在密码学中），或者是假想的（如在去随机化中），但是无论何种情况，都希望能阻止其看到某个事物或做某件事，而困难问题有助于达成此目标。

假设 $P \neq NP$ 或者甚至 NP 不包含在 BPP 中。这样的假设有用吗？不一定： $NP \not\subseteq BPP$ 的假设针对着问题在最坏情况下的复杂性，而要让困难问题起作用似乎必须具备生成困难实例的能力。特别地，生成的实例应该是典型困难的，而不是仅具有偶然情况下的困难性。就是说，我们追求的是平均情况下的困难性而不仅是最坏情况下的困难性。

稍微偏离一下正题，在 7.2 节将看到，在 ε 类中，最坏情况下的困难性（ NP 类或者甚至 ε 类）可以转化成平均情况下的困难性。这种转化在 NP 类中是否成立尚不明确，但在某些应用（如密码学）中，人们确实希望平均情况下困难的问题属于 NP 。此时，需要假设对于 NP 类中某些问题，困难实例是易于生成的（而不仅仅是存在）。也就是说，假设 NP 类关于某种可高效采样的分布具有“平均情况下的困难性”。10.2 节将进一步讨论这种假设。

然而这种假设对于上述应用（如密码学）似乎还不够：只有当我们能高效地生成困难实例以及足够多的解时，^①才能利用这些“平均情况下困难”的问题。换言之，假设对于 PC 中某些搜索问题（或 NP 中某些判定问题），可以高效生成“实例—解”构成的对（或“是”实例及相应的 NP -证据），使得该实例是难解的（或难以验证是否为集合成员）。毫无疑问，困难假设针对的是那些无法得到解（或证据）的人。因此，可以高效地生成“难题”及解，然后把这只有生成者知道解的难题呈给其他人。

现在对上述讨论进行形式化定义。参考定义 2.3，考虑 PC 中的关系 R （即 R 为多项式界限且 R 的成员可在多项式时间内判定），并假设存在概率多项式时间算法 G ，满足以下两个条件。

(1) 对于输入 1^n ，算法 G 总能生成 R 中的一个对，其中第一个元素长度为 n ，即 $\Pr[G(1^n) \in R \cap (\{0,1\}^n \times \{0,1\}^*)] = 1$ 。

(2) 对于由 G 生成的实例求解是不可行的，即当只给定 $G(1^n)$ 的第一个元素时，不可能求解。将 $G(1^n)$ 的第一个元素记作 $G_1(1^n)$ ，对于任意概率多项式时间算法 S （求解算法），有 $\Pr[(G_1(1^n), S(G_1(1^n))) \in R] = \mu(n)$ ，其中 μ 比任意多项式的倒数衰减速度快（即对任意正多项式 p 和所有足够大的 n ，有 $\mu(n) < 1/p(n)$ ）。

称 G 为 R 的已解困难实例生成器。以下证明当且仅当单向函数存在时这种生成器存在。这里的单向函数是指易求值但（在平均意义上）难求逆的函数。也就是说，函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 是单向的，如果存在一个高效算法，对于输入 x 能输出 $f(x)$ ，而任何一个试图求 $f(x)$ 原像的算法其成功概率可以忽略（这里的概率空间是对所有可能的 x 以及算法的随机数）。结合可行计算、概率多项式时间算法、可忽略的函数以及衰减速度快于任何多项式倒数的函数，可以得到如下定义。

定义 7.1（单向函数） 函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 称为单向函数，如果以下两个条件满足：

(1) 易求值。存在多项式时间算法 A ，使得对任意 $x \in \{0,1\}^*$ ，有 $A(x) = f(x)$ 。

^① 要强调这里讨论的两个缝隙之间的差别。我们认为从直观上看，最坏情况下的困难性比平均情况下的困难性更加无用，后者并不要求高效生成“已解”实例。

(2) 难求逆。对任意概率多项式时间算法 A' , 任意多项式 p , 以及足够大的 n , 有

$$\Pr_{x \in \{0,1\}^n} [A'(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)} \quad (7.1)$$

其中概率空间是所有 $x \in \{0,1\}^n$ 以及算法 A' 内部随机数的所有可能取值。

对算法 A' 给定一个辅助输入 1^n , 从而使其运行时间为 x 长度的多项式, 当 f 对输入产生严重压缩时 (如 $|f(x)| = O(\log|x|)$), 这一点很重要。典型地 (并且事实上, 不失一般性, 见习题 7.1), 函数 f 能保持长度, 此时, 辅助输入 1^n 是多余的。注意: 并不要求 A' 输出 $f(x)$ 的某个特定原像, 任意原像 (即集合 $f^{-1}(f(x))$ 中元素) 都可满足要求 (事实上, 当 f 为双射时, 串 x 是 $f(x)$ 唯一的原像, 但在一般情况下, 可能存在其他原像)。定义中要求算法 A' 以压倒性优势失效 (寻找原像), 这里概率空间为对所有输入求值。即 f 是“典型地”难以求逆, 而不是仅仅对某些 (“少数”) 情况难求逆。

命题 7.2 以下两个条件等价。

(1) 对于某些 $R \in \mathcal{NP}$, 存在已解困难实例生成器。

(2) 单向函数是存在的。

证明概要: 设 G 为某个 $R \in \mathcal{NP}$ 的已解困难实例生成器, 并假设对于输入 1^n , G 产生 $l(n)$ 个随机数。为叙述简洁, 假设 $l(n) = n$, 并考虑函数 $g(r) = G_1(1^n, r)$, 其中 $G(1^n, r)$ 表示 G 对于输入 1^n 利用随机数 r 生成的输出 (G_1 的定义如上), 则 g 必为单向函数, 因为 g 的求逆算法对于输入 $x = g(r)$ 将得到 r' , 使得 $G_1(1^n, r') = x$, 而 $G(1^n, r')$ 一定属于 R (意为 $G(1^n, r')$ 的第二个元素是 x 的一个解)。当 $l(n) \neq n$ 时 (并不失一般性, 假设 $l(n) \geq n$), 我们定义 $g(r) = G_1(1^n, s)$, 其中 n 为满足 $l(n) \leq r$ 的最大整数, 而 s 为 r 的 $l(n)$ 比特前缀。

另一方面, 假设 f 为单向函数 (且 f 保持长度)。考虑算法 $G(1^n)$: 均匀选择 $r \in \{0,1\}^n$, 并输出 $(f(r), r)$ 。令 $R \stackrel{\text{def}}{=} \{(f(x), x) : x \in \{0,1\}^*\}$, 则 R 属于 $\mathcal{PC}$, 而 G 为 R 的已解困难实例生成器, 因为 R 的任意求解算法 (对于由 G 生成的实例) 可以高效地对 f 关于 $f(U_n)$ 求逆。□

说明: 附录 C.2.1 中介绍了几种单向函数以及基本定义的变形。这里为了将来讨论方便, 定义更强意义上的单向函数, 其含义是指由多项式规模的非一致性电路对函数求逆是不可行的。我们使用另一种定义, 建立于附录 D.1.1 中的概率论基础之上。^①

定义 7.3 (单向函数, 非一致的困难) 单向函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 称为非一致难求逆的, 如果对任意多项式规模的电路簇 $\{C_n\}$, 任意多项式 p , 以及所有足够大的 n , 有

$$\Pr[C_n(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

注意: 如果一个函数由多项式规模电路求逆是不可行的, 则它也很难用概率多项式时间算法求逆, 就是说当缺乏随机性时, 可以用非一致性 (以及其他条件) 来补偿, 见习题 7.2。

① 具体地, 令 U_n 表示在 $\{0,1\}^n$ 上均匀分布的随机变量, 可将式 (7.1) 写成 $\Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < 1/p(n)$, 回顾 U_n 的两种表现形式都涉及同一种采样。

7.1.2 弱单向函数的放大

以上讨论解释了较强意义上的“平均困难性”。具体而言,要求任意可行的算法几乎总是(即除了以可忽略的概率成功外)无法求解问题(如对单向函数求逆)。这种解释实际上适合于各种应用。然而,平均情况下的困难性有一种较弱解释也很常用,它只要求任意可行的算法通常(即以较明显的概率)无法求解问题。本节的主要目标是要说明较温和的平均困难性可以转化为 7.1.1 节中讨论的较强形式。首先利用单向函数的框架来定义温和的平均困难性。具体地,定义弱单向函数。

定义 7.4 (弱单向函数) 函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 称为弱单向的,如果满足以下两个条件。

(1) 易求值。如定义 7.1。

(2) 求逆的难度较小。存在一个正多项式 p , 使得对任意概率多项式时间算法 A' 及所有足够大的 n , 有

$$\Pr_{x \in \{0,1\}^n} [A'(f(x), 1^n) \notin f^{-1}(f(x))] > \frac{1}{p(n)} \quad (7.2)$$

其中概率空间是所有可能的 $x \in \{0,1\}^n$ 以及算法 A' 的内部随机数。此时,称 f 是 $1/p$ 单向的。

这里要求算法 A' 有明显的失效(对于函数 f 一个随机的像,无法找到其原像)概率,而不是以压倒性的高概率失效(如定义 7.1)。为了更清晰地表述,偶而也使用定义 7.1 中的单向函数,此时称其为强单向函数。

注意: 假设单向函数存在,则存在着非强单向函数的弱单向函数(见习题 7.3)。然而,任意弱单向函数可以转化为强单向函数,这是本小节的主要结论。

定理 7.5 (单向函数的放大) 弱单向函数的存在性蕴含着强单向函数的存在性。

证明概要: 用直接构造法,只需要将新的函数分为足够多的块,并在每一块上应用弱单向函数。即对某个多项式 p , 假设 f 是 $1/p$ 单向的,考虑如下函数

$$F(x_1, x_2, \dots, x_t) = (f(x_1), f(x_2), \dots, f(x_t)) \quad (7.3)$$

其中 $t \stackrel{\text{def}}{=} n \cdot p(n)$, 而 $x_1, x_2, \dots, x_t \in \{0,1\}^n$ 。

事实上 F 可以扩展到 $\{n^2 \cdot p(n) : n \in \mathbf{N}\}$ 之外的串,而这种扩展对所有原像也是难求逆的。^①

提醒读者注意,对函数 F 求逆的困难性并不是仅仅建立在“组合”(即考虑对任意 $S \subset \{0,1\}^n$, 相关的块 S' 属于 $(\{0,1\}^n)^t$, 其中 S 表示“易求逆”的 f 的原像集合)的原因之上。特别地,不能假设求逆算法在每一块中独立地工作。这个假设看似合理,但我们还不知道是否遗漏了什么(实际上,证明这种假设没有任何遗漏是一个极难的研究课题)。一般情况下,不能针对所有高效算法做出假设(在单向函数的定义下),除非确实可以证明这些假设没有任何遗漏。

证明对函数 F 求逆的困难性时,使用一个所谓的可约性方法 (Reducibility Argument) (可用于证明该领域中所有的有条件结论)。可约性方法实际上是一种归约,但对其分析是基于平

^① 一种简单的扩展是定义 $F(x) = F(x_1, x_2, \dots, x_{n \cdot p(n)})$, 其中 n 为满足 $n^2 p(n) \leq |x|$ 的最大整数, x_i 为 x 中第 i 个连续的 n 比特串 (即 $x = x_1 x_2 \dots x_{n \cdot p(n)} x'$, 其中 $x_1 x_2 \dots x_{n \cdot p(n)} \in \{0,1\}^{n^2 p(n)}$)。

均情况复杂性。具体来讲,我们将证明从任意一个能以不可忽略的成功概率对 F 求逆的算法出发,可以构造对原始函数 f 求逆的算法,其成功概率与假设不同(关于 f)。换句话说,将对 f 的“强求逆”任务(即违背其弱单向性)归约为对 F 的“弱求逆”任务(即违背其强单向性)。特别地,对于输入 $y = f(x)$,归约中多次调用对 F 求逆的函数(多项式次),每次调用时,向其输入一个随机的 f 的像序列,并且 y 随机出现在其中的某个位置(事实上,这个序列对应着 F 的随机像)。具体细节如下。

用反证法。假设 F 不是强单向的,即存在概率多项式时间算法 B' 和多项式 $q(\cdot)$,使得对于无限多个 m ,有

$$\Pr[B'(F(U_m)) \in F^{-1}(F(U_m))] > \frac{1}{q(m)} \quad (7.4)$$

主要讨论 m 的一般情况,假设 $m = n^2 p(n)$,构造如下的概率多项式时间算法 A' 来对 f 求逆。对于输入 y 和 1^n (对某个 $x \in \{0,1\}^n$,假设 $y = f(x)$),算法 A' 对于输入 y ,进行如下的概率计算,记作 I ,共迭代 $t'(n)$ 次,其中 $t'(\cdot)$ 是一个与多项式 p 和 q 相关的多项式(具体地,令 $t'(n) \stackrel{\text{def}}{=} 2n^2 \cdot p(n) \cdot q(n^2 p(n))$)。

程序 I (输入 y 和 1^n)

从 $i=1$ 到 $t'(n) \stackrel{\text{def}}{=} n \cdot p(n)$

(1) 均匀独立地选择串序列 $x_1 x_2 \cdots x_{t(n)} \in \{0,1\}^n$ 。

(2) 求 $(z_1, z_2, \dots, z_{t(n)}) \leftarrow B'(f(x_1), f(x_2), \dots, f(x_{i-1}), y, f(x_{i+1}), f(x_{i+2}), \dots, f(x_{t(n)}))$ (注意:在第 i 个位置用 y 来代替 $f(x_i)$)。

(3) 如果 $f(z_i) = y$,则停机,输出 z_i (此时,认为算法已成功)。

结束。

根据式 (7.4),可以求出算法 A' 的成功概率下限,而这与定理的假设相矛盾。为此,定义一个集合 S_n ,其中包含使程序 I 成功概率大于 $n/t'(n)$ 的所有 n 比特串(概率空间为 I 的所有随机输入),即

$$S_n \stackrel{\text{def}}{=} \left\{ x \in \{0,1\}^n : \Pr[I(f(x)) \in f^{-1}(f(x))] > \frac{n}{t'(n)} \right\}$$

在以下两个断言中,我们将证明 S_n 至多包含长为 n 的串中的 $1/2p(n)$,并且对每个串 $x \in S_n$,算法 A' 对 $f(x)$ 成功求逆的概率指数地接近 1,从而得出 A' 对 $f(U_n)$ 成功求逆的概率大于 $1 - (1/p(n))$,与定理假设矛盾。

断言 7.5.1 对任意 $x \in S_n$,有 $\Pr[A'(f(x)) \in f^{-1}(f(x))] > 1 - 2^{-n}$ 。

这个断言由 S_n 和 A' 的定义直接可证。

断言 7.5.2 $|S_n| > \left(1 - \frac{1}{2p(n)}\right) \cdot 2^n$ 。

定理证明中剩余的部分就是证明断言 7.5.2, 事实上, 结合断言 7.5.1 与断言 7.5.2, 可以证明定理 7.5。

一个重要的观察是, 对任意 $i \in [t(n)]$ 及任意 $x_i \in \{0,1\}^n \setminus S_n$, 有

$$\begin{aligned} & \Pr \left[B' \left(F \left(U_{n^2 p(n)} \right) \right) \in F^{-1} \left(F \left(U_{n^2 p(n)} \right) \right) \mid U_n^{(i)} = x_i \right] \\ & \leq \Pr \left[I(f(x_i)) \in f^{-1}(f(x_i)) \right] \leq \frac{n}{t'(n)} \end{aligned}$$

其中 $U_n^{(1)}, U_n^{(2)}, \dots, U_n^{(n \cdot p(n))}$ 表示随机变量 $U_{n^2 p(n)}$ 中的 n 比特块, 从而有

$$\begin{aligned} \xi & \stackrel{\text{def}}{=} \Pr \left[B' \left(F \left(U_{n^2 p(n)} \right) \right) \in F^{-1} \left(F \left(U_{n^2 p(n)} \right) \right) \wedge (\exists i, \text{使得 } U_n^i \in \{0,1\}^n \setminus S_n) \right] \\ & \leq \sum_{i=1}^{t(n)} \Pr \left[B' \left(F \left(U_{n^2 p(n)} \right) \right) \in F^{-1} \left(F \left(U_{n^2 p(n)} \right) \right) \wedge U_n^i \in \{0,1\}^n \setminus S_n^* \right] \\ & \leq t(n) \cdot \frac{n}{t'(n)} = \frac{1}{2q(n^2 p(n))} \end{aligned}$$

其中不等式成立是由于 $t'(n) = 2n^2 \cdot p(n) \cdot q(n^2 p(n))$ 以及 $t(n) = n \cdot p(n)$ 。另一方面, 利用式 (7.4), 有

$$\begin{aligned} \xi & \geq \Pr \left[B' \left(F \left(U_{n^2 p(n)} \right) \right) \in F^{-1} \left(F \left(U_{n^2 p(n)} \right) \right) - \Pr \left[(\forall i) U_n^{(i)} \in S_n \right] \right] \\ & \geq \frac{1}{q(n^2 p(n))} - \Pr[U_n \in S_n]^{t(n)} \end{aligned}$$

根据 $t(n) = n \cdot p(n)$, 有 $\Pr[U_n \in S_n] > (1/2q(n^2 p(n)))^{1/(n \cdot p(n))}$, 这蕴含了对足够大的 n , 有 $\Pr[U_n \in S_n] > 1 - (1/2p(n))$, 从而证明了断言 7.5.2, 于是定理得证。□

思考

回顾定理 7.5 的证明结构。给定一个弱的单向函数 f , 首先构造一个多项式时间可计算的函数 F , 然后再证明 F 是强单向的。为证明 F 为强单向的, 利用了可约性方法, 将一个可能与 F 的强单向性假设矛盾的高效算法转化为与 f 的弱单向性假设矛盾的高效算法, 从而 F 一定是强单向的。我们强调这里的转化算法实际上是一个随机 Cook-归约, 对 F 求逆算法的结构没有任何假设。这类假设 (如 “自然地” 假设 F 的求逆算法独立地作用于每一块) 可能并不合理 (至少在目前所理解的高效计算并不合理)。

使用可约性方法的术语, 而不是仅仅称其为归约, 是为了强调没有使用标准 (最坏情况复杂性) 归约。要澄清两者的区别: 两种归约都是将求解一个问题转化为求解另一个问题, 即使用一个求解后者的程序来构造求解前者的程序。然而, 在标准归约中, 人们假设存在解第二个问题的完美程序, 可对任意实例求解 (即在最坏情况下), 并用该程序构造解第一个问题的程序。因此, 归约中可以对极其 “非典型” 的实例调用给定程序 (来解第二个问题), 而这在可约性方法中是不允许的。其中给定的解第二个问题的算法只能关于某种分布, 以某个概率求解。因此, 在使用可约性方法时, 并不能对任意实例调用该算法。相反, 必须考虑由归约所诱导的关于第二个问题实例的概率分布。此时 (以及许多其他情况下), 这种分布就是

假设中（考虑第二个问题可解时）所指的分布，但是一般来说这些分布不需要满足“充分的封闭性”（这依赖于具体地分析）。在任何情况下，仔细考虑由可约性方法所诱导的分布是必须的（事实上，同样的问题也出现在 10.2 节中讨论的“分布问题”之间的归约中）。

信息论中的类似结论。定理 7.5（或其证明过程）在信息论（或概率论）中有一个自然的类似结论，其中涉及到通过重复实验而使成功概率放大：如果某事件在单次实验中发生的概率为 p ，则当重复该实验 n/p 次后，该事件会以非常高的概率（即 $1 - e^{-n}$ ）发生。与定理 7.5 的相似之处是指对于随机输入求函数 F 的值，这里输入中的每一块可以看作是尝试对 f 中明显的“困难区域”求逆。读者此时也许能够确信，定理 7.5 的证明比信息论中类似结论的证明要更复杂。在信息论中，重复的实验被定义为是相互独立的，而在计算理论中，不能保证这种独立性（事实上，独立的假设对应于定理 7.5 证明中一开始的简单情形）。两种环境的另一不同之处是：在信息论环境中，事件不发生的概率将随着重复次数的增加而呈指数降低；相反，在计算环境中，多项式时间算法求逆的成功概率只能到达一个未明确定义的可忽略的界限。进一步地，据我们所知，也许函数 F 在 $F(U_{n^2 p(n)})$ 处可以高效求逆，其成功概率呈亚指数降低（如概率为 $2^{-(\log_2 n)^3}$ ），而信息论中的类似结论则呈指数降低（即 e^{-n} ）（图 7.1）。

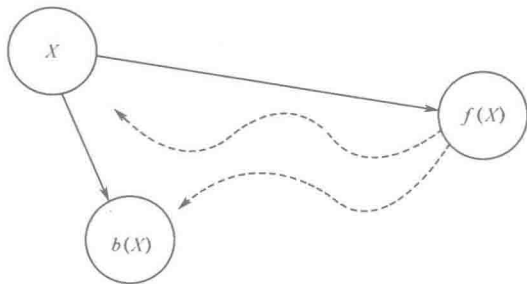


图 7.1 单向函数的困难核心。实心箭头表示易于计算的转换，虚线箭头表示不可行的转换

7.1.3 困难核心谓词

单向函数本身足以实现一个重要应用：构造安全的签名方案（见附录 C.6）。在其他应用中，则不仅需要利用单向函数完全恢复原像的不可行性，还要利用有意义地猜测原像中比特的不可行性。后一个概念被定义为困难核心谓词。

回顾前面讲到函数 f 是单向的意味着给定一个 y （在 f 的值域中），找到 y 关于 f 的原像是不可行的。但这并不意味着不可能得到有关 y 的原像的部分信息。特别地，可能恢复原像中一半的比特是容易的（如给定一个单向函数 f ，对任意 $|x| = |r|$ ，考虑函数 f' ，定义为 $f'(x, r) \stackrel{\text{def}}{=} (f(x), r)$ ）。注意：（函数的原像中）被隐藏的部分信息在更高级的构造（如构造伪随机数生成器或安全的加密方案）中起着重要作用。由此出发，我们将证明任意单向函数在本质上都隐藏了其原像中特定的部分信息，而这些部分信息由原像本身是易求的。这些部分信息可以看作是对 f 求逆的“困难核心”（Hard-core）。不严格地说，一个多项式时间可计算的（布尔）谓词 b 称为函数 f 的困难核心，如果不存在可行的算法，在给定 $f(x)$ 时，能以大于 $1/2$ 的概率成功猜测出 $b(x)$ 。

定义 7.6 (困难核心谓词) 一个多项式时间可计算的谓词 $b: \{0,1\}^* \rightarrow \{0,1\}$ 称为函数 f 的困难核心, 如果对任意概率多项式时间算法 A' , 任意正多项式 $p(\cdot)$ 以及所有足够大的 n , 有

$$\Pr[A'(f(x)) = b(x)] < \frac{1}{2} + \frac{1}{p(n)}$$

其中概率空间是随机均匀选择的所有 $x \in \{0,1\}^n$ 以及算法 A' 所有可能的内部随机数。

注意: 对任意 $b: \{0,1\}^* \rightarrow \{0,1\}$ 以及 $f: \{0,1\}^* \rightarrow \{0,1\}^*$, 存在一些明显的算法能由 $f(x)$ 猜测 $b(x)$, 猜测成功的概率至少为 $1/2$ (例如, 算法可以从其输入中, 均匀选择一个比特输出即可), 并且如果 b 是 (任意函数的) 困难核心谓词, 则 b 几乎是无偏移的 (即对于均匀选择的 x , 概率差 $|\Pr[b(x)=0] - \Pr[b(x)=1]|$ 必为关于 n 的可忽略函数)。

由于 b 本身可在多项式时间内计算, 不存在高效算法 (以大于 $1/2$ 的成功概率) 由 $f(x)$ 求 $b(x)$ 的原因或者是由于 f 的信息丢失 (即 f 不是双射), 或者是由于对 f 求逆的困难性。例如, 对于 $\sigma \in \{0,1\}$ 和 $x' \in \{0,1\}^*$, 谓词 $b(\sigma x') = \sigma$ 是函数 $f(\sigma x') \stackrel{\text{def}}{=} 0x'$ 的困难核心。因此, 在这种情况下, b 成为 f 的困难核心是由于在计算 f 时有信息丢失 (具体来讲, 就是第一个比特 σ)。另一方面, 如果 f 在计算中没有信息丢失 (即 f 为双射), 则 f 存在困难核心只可能因为 f 是难求逆的。大体上, 一个有趣的事实是困难核心是一种计算现象而非信息论现象 (由于 f 的“信息丢失”)。因而, 任意存在修正版本的单向函数都会有困难核心谓词。

定理 7.7 (一类通用的困难核心谓词) 对任意单向函数 f , x 与 r 模 2 的内积, 记作 $b(x, r)$, 是 $f'(x, r) = (f(x), r)$ 的困难核心。

换言之, 定理 7.7 称, 给定 $f(x)$ 和随机子集 $S \subseteq [x]$, 以大于 $1/2$ 的概率猜测 $\bigoplus_{i \in S} x_i$ 是不可行的, 其中 $x = x_1 x_2 \cdots x_n$ 在 $\{0,1\}^n$ 上均匀分布。

证明概要: 证明使用可约性方法 (见 7.1.2 节)。具体而言, 将对 f 求逆的任务归约为求 f' 的困难核心, 并确保归约 (应用于服从求逆任务中分布的输入时) 中生成的分布与谓词定义中的分布相同。因此, 如果 b 不是 f' 的困难核心, 则将得到与 f 难求逆的假设相矛盾的结论。我们强调这个讨论远比分析相应的“概率论”情形 (即 $(r, b(X, r))$ 的分布, 其中 $r \in \{0,1\}^n$ 为均匀分布, X 是一个随机变量, 具有超对数的最小熵 (表示给定 $f(x)$ 时, 关于 x 的“有效”知识)^①) 复杂得多。

证明的出发点是一个概率多项式时间算法 B , 满足对某个多项式 p 以及无穷多个 n , 有 $\Pr[B(f(X_n), U_n) = b(X_n, U_n)] > \frac{1}{2} + \frac{1}{p(n)}$, 其中 X_n 和 U_n 在 $\{0,1\}^n$ 上均匀、独立分布。利用一个简单的平均情况, 主要讨论 $\varepsilon = 1/2p(n)$ 时, 占 x 取值范围中比例 ε 且使 $\Pr[B(f(x), U_n) = b(x, U_n)] > \frac{1}{2} + \varepsilon$ 成立的 x 。假设 x 属于这个良好定义的集合 (密度为 ε), 对于输入的 $f(x)$, 我们将指出如何利用 B 来对 f 求逆。

① X 的最小熵定义为 $\min_v \{\log_2(1/\Pr[X=v])\}$, 也就是说, 如果 X 的最小熵为 m , 则 $\max_v \{\Pr[X=v]\} = 2^{-m}$ 。剩余 Hash 引理 (见附录 D.2) 表明, 在这种情况下, $\Pr[b(X, U_n) = 1 | U_n] = \frac{1}{2} \pm 2^{-\Omega(m)}$, 其中 U_n 表示 $\{0,1\}^n$ 上的平均分布。

作为热身,先暂时假设对于上述 x ,算法 B 的成功概率 $p > \frac{3}{4} + 1/\text{poly}(|x|)$ 而不是 $p > \frac{1}{2} + 1/\text{poly}(|x|)$ 。此时,由 $f(x)$ 恢复 x 是很容易的:为了求 x 的第 i 比特,记作 x_i ,只需随机选择 $r \in \{0,1\}^{|x|}$,并得到 $B(f(x), r)$ 和 $B(f(x), r \oplus e^i)$,其中 $e^i = 0^{i-1}10^{n-i}$,而 $v \oplus u$ 表示二元向量 v 和 u 模2加。上述方案及定理证明剩余的部分的关键是 $b(x, r \oplus s) = b(x, r) \oplus b(x, s)$,这可以很容易地由 $b(x, y) = \sum_{i=1}^n x_i y_i \pmod{2}$ 来验证,并注意到两个比特的模2加对应着其XOR(异或运算)。现在,如果 $B(f(x), r) = b(x, r)$ 和 $B(f(x), r \oplus e^i) = b(x, e^i)$ 均成立,则 $B(f(x), r) \oplus B(f(x), r \oplus e^i)$ 等于 $b(x, r) \oplus b(x, r \oplus e^i) = b(x, e^i) = x_i$ 。对于随机选择的 r , $B(f(x), r) = b(x, r)$ 和 $B(f(x), r \oplus e^i) = b(x, r \oplus e^i)$ 均成立的概率至少是 $1 - 2 \cdot (1 - p) > \frac{1}{2} + \frac{1}{\text{poly}(|x|)}$ 。因此,重复上述步骤足够多次之后(利用随机独立选择的 r),并根据大数原则,可以以非常高的概率恢复 x_i 。用类似方法可以恢复 x 中所有比特,从而对 f 在 $f(x)$ 处求逆。然而,上述所有分析是建立在(无法证实的)假设 $p > \frac{3}{4} + 1/\text{poly}(|x|)$ 之上的,而我们只知道 $p > \frac{1}{2} + \varepsilon$,对某个 $\varepsilon = 1/\text{poly}(|x|)$ 。

上述过程存在的问题是使算法 B 在计算形如 $(f(x), \cdot)$ 的输入时,错误概率加倍。在(上述)不现实的假设下,即 B 对于这类输入的平均错误概率远远小于 $1/4$,"错误加倍"现象不会有任何问题。然而,在一般情况下(甚至当 B 的错误概率恰好为 $1/4$ 时),上述算法不太可能会对 f 成功求逆。注意: B 的平均错误概率(对某个固定的 $f(x)$,并对随机选择的 r 求平均值)不能通过重复执行来减少(如对每个 x , B 对于输入 $(f(x), r)$ 做出正确回答的概率总为 $3/4$,而做出错误回答的概率总为 $1/4$)。所以这里要求用其他方式应用算法 B ,使 B 的原始错误概率不会加倍。

其中的关键思想是改变生成 r 的方法,使得对每个 r (和 i)只允许应用算法 B 一次,而不是两次。具体地,对 $(f(x), r \oplus e^i)$ 调用 B ,得到对 $b(x, r \oplus e^i)$ 的猜测,并以不同方式得到 $b(x, r)$ (不需要使用 B)。这样做的优点是错误概率不会加倍,因为只利用 B 来得到对 $b(x, r \oplus e^i)$ 的"猜测"。然而,缺点是还需知道 $b(x, r)$,而我们尚不清楚如何才能不用 B 来得到 $b(x, r)$,只能猜测 $b(x, r)$ 的值。如果只需要对一个 r (或关于 $|x|$ 的对数多个 r)猜测 $b(x, r)$,这样做尚可,但当需要对多项式个 r 猜测 $b(x, r)$ 的值时,问题就出现了。猜测这些 $b(x, r)$ 的平凡方法具有指数级的小成功概率。然而,如果在生成这些多项式数量的 r 时,使得一方面,它们具有"足够的随机性",另一方面,可以以不可忽略的成功概率猜测所有 $b(x, r)$ 的值^①。特别地,使生成的 r 是两两独立的,从而可以满足这两个(矛盾的)要求。我们强调此时如果成功(猜测了所有的 $b(x, r)$),则可以以很高的概率恢复 x 。这样就可以以不可忽略的概率恢复出了 x 。

现在需要说明如何生成两两独立的 r (从而猜测相应的 $b(x, r)$)。为了生成 $m = \text{poly}(|x|)$ 个 r ,均匀(且独立地)选择 $\{0,1\}^{|x|}$ 中的 $l \stackrel{\text{def}}{=} \log_2(m+1)$ 个串,并将这些串记作 $s^1, s^2, \dots, s^l$ 。然后,

① 另一种方法是尝试所有多项式数量的猜测。此时,将输出一张表,其中以很高概率包含 x (见习题7.6)

猜测 $b(x, s^l)$ 到 $b(x, s^l)$ 的值, 将这些从 $\{0, 1\}$ 中均匀 (独立) 选择的猜测记作 σ^l 到 σ^l , 从而成功猜测出所有 $b(x, s^i)$ 的概率为 $2^{-l} = \frac{1}{\text{poly}(|x|)}$ 。不同的 r 对应于 $\{1, 2, \dots, l\}$ 的不同非空子集。具

体地, 对每个这样的子集 J , 令 $r^J \stackrel{\text{def}}{=} \bigoplus_{j \in J} s^j$ 。易验证这些 r^J 是两两独立的, 且在 $\{0, 1\}^{|x|}$ 中均匀分布, 见习题 7.5。关键在于 $b(x, r^J) = b(x, \bigoplus_{j \in J} s^j) = \bigoplus_{j \in J} b(x, s^j)$ 。因此, 对 $b(x, r^J)$ 的猜测即为 $\bigoplus_{j \in J} \sigma^j$, 且所有猜测都成功的概率是不可忽略的。综合上述所有讨论, 可以得到以下的算法, 其中 $\varepsilon = 1/\text{poly}(n)$ 表示对于占比例 ε 的 x , B 猜测 $b(x, \cdot)$ 的优势下界 (即对这些 x , 有 $\Pr[B(f(x), U_n) = b(x, U_n)] > \frac{1}{2} + \varepsilon$)。

求逆算法 (对于输入 $y = f(x)$ 及参数 n 和 ε):

令 $l = \log_2(n/\varepsilon^2) + O(1)$

(1) 均匀独立地选择 $s^1, s^2, \dots, s^l \in \{0, 1\}^n$, 均匀独立地选择 $\sigma^1, \sigma^2, \dots, \sigma^l \in \{0, 1\}$ 。

(2) 对每个非空子集 $J \in [l]$, 计算 $r^J = \bigoplus_{j \in J} s^j$ 和 $\rho^J = \bigoplus_{j \in J} \sigma^j$ 。

(3) 对 $i = 1, 2, \dots, n$, 根据多数原则, 在由比特 $(\rho^J \oplus B(f(x), r^J \oplus e^i))_{\emptyset \neq J \subseteq [l]}$ 构成的长为 $(2^l - 1)$ 的串中, 确定比特 z_i 。

(4) 输出 $z_1, z_2, \dots, z_n$ 。

注意: 第 (3) 步中的“表决方案”使用了两两独立的样本 (即各个 r^J), 但是其工作方式本质上似乎工作于独立的采样之上 (即独立的 r)。^① 也就是说, 对所有的 i 及 J , 有 $\Pr_{s^1, s^2, \dots, s^l} [B(f(x), r^J \oplus e^i) = b(x, r^J \oplus e^i)] > (1/2) + \varepsilon$, 其中 $r^J = \bigoplus_{j \in J} s^j$, 且 (对每个 i) 对应于不同 J 的事件是两两独立的。从而, 如果对任意 $j \in [l]$, 有 $\sigma^j = b(x, s^j)$, 则对每个 i 及 J , 有

$$\begin{aligned} & \Pr_{s^1, s^2, \dots, s^l} [\rho^J \oplus B(f(x), r^J \oplus e^i) = b(x, e^i)] \\ &= \Pr_{s^1, s^2, \dots, s^l} [B(f(x), r^J \oplus e^i) = b(x, r^J \oplus e^i)] > \frac{1}{2} + \varepsilon \end{aligned} \quad (7.5)$$

其中等式成立是由于 $\rho^J = \bigoplus_{j \in J} \sigma^j = b(x, r^J) = b(x, r^J \oplus e^i) \oplus b(x, e^i)$ 。注意: 式 (7.5) 针对着单次表决 $b(x, e^i)$ 的正确性。利用 $m = 2^l - 1 = O(n/\varepsilon^2)$, 并注意到这些 (布尔) 表决是两两独立的, 可以推断出大部分表决出错的概率上限为 $1/2n$ 。对所有 i 求联合概率上限, 可以推导出大多数表决正确的概率至少为 $1/2$, 因此可以正确恢复 x 。回顾上述推理成立的条件是对每

① 这里的关键是由样本中得到的近似值的正确性, 而非错误概率。我们希望在求 $\Pr[b(x, r) \oplus B(f(x), r \oplus e^i) = 1]$ 的近似值时, 误差项为 ε , 因为这样的近似值可以正确地确定 $b(x, e^i)$ 。在求 $[0, 1]$ 中某个值的近似时, 利用 $O(t/\varepsilon^2)$ 个两两独立的样本点, 可以在增加一个 ε 项时, 将错误概率控制在 $1/t$, 而利用完全随机的同样多样本, 得到的错误概率为 $\exp(-t)$ 。由于可令 $t = \text{poly}(n)$, 并使错误概率为 $1/2n$, 所以两个近似方案的错误概率差值并不大。进一步的讨论可见于附录 D.1.2 和附录 D.3。

个 $j \in [l]$, 有 $\sigma^j = b(x, s^j)$, 而这个事件成立的概率是 $2^{-l} = (m+1)^{-1} = \Omega(\varepsilon^2/n) = 1/\text{poly}(n)$ 。

因此, 正确恢复 x 的概率是 $1/\text{poly}(n)$, 定理得证。□

思考

观察定理 7.7 的证明, 注意到, 它实际上使用了一个对 $b(x, \cdot)$ 求近似值的黑盒子 $B_x(\cdot)$, 具体而言, 在定理 7.7 中, $B_x(r) \stackrel{\text{def}}{=} B(f(x), r)$ 。特别地, 证明中没有利用可以验证由上述过程恢复的原像的正确性这一事实。因此, 证明过程实际上表明存在一个 $\text{poly}(n/\varepsilon)$ -时间预言机, 对任意 $x \in \{0, 1\}^n$, 在给定到任意满足

$$\Pr_{r \in \{0, 1\}^n} [B_x(r) = b(x, r)] \geq \frac{1}{2} + \varepsilon \quad (7.6)$$

的 $B_x: \{0, 1\}^n \rightarrow \{0, 1\}$ 的预言访问时, 以至少 $\text{poly}(n/\varepsilon)$ 的概率输出 x 。具体地, 输出 x 的概率至少为 $p \stackrel{\text{def}}{=} \Omega(\varepsilon^2/n)$ 。注意: x 仅仅是一个使式 (7.6) 成立的串, 从而至多有 $1/p$ 个串满足式 (7.6)。进一步地, 通过将上述过程迭代 $\tilde{O}(1/p)$ 次, 可以得到所有这些串 (见习题 7.7)。

定理 7.8 (重述定理 7.7) 存在一个概率预言机, 给定参数 n, ε 及对任意函数 $B: \{0, 1\}^n \rightarrow \{0, 1\}$ 的预言访问时, 在 $\text{poly}(n/\varepsilon)$ 步后停机, 并以至少 $1/2$ 的概率输出所有串 $x \in \{0, 1\}^n$ 的列表, 满足

$$\Pr_{r \in \{0, 1\}^n} [B(r) = b(x, r)] \geq \frac{1}{2} + \varepsilon \quad (7.7)$$

其中 $b(x, r)$ 表示 x 与 r 模 2 的内积。

可以修改该机器使其输出列表以极大概率不包含任何满足 $\Pr_{r \in \{0, 1\}^n} [B(r) = b(x, r)] < \frac{1}{2} + \frac{\varepsilon}{2}$ 的串 x 。

定理 7.8 的含义是, 如果给定关于 x 的某种信息时, 恢复 x 是困难的, 则给定同样信息 & 一个随机数 r 时, 预测 $b(x, r)$ 也是困难的。这个断言可用反证法 (见习题 7.14) 证明。^①事实上, 这种说法本身也体现了定理 7.7 的实质, 只是其针对着任意“关于 x 的信息” (而不是 $f(x)$ 的值)。为了论证这一点, 可将上述解释重述为: 对任意随机过程 Π , 如果给定 s 时, 得到 $\Pi(s)$ 是困难的, 则给定 s 及一个均匀分布的 $r \in \{0, 1\}^{|\Pi(s)|}$ 时, 预测 $b(\Pi(s), r)$ 也是困难的。^②

来自编码理论的解释

定理 7.8 可以看作是 Hadamard 码的“查表译码”程序。串 $x \in \{0, 1\}^n$ 的 Hadamard 编码是一个 2^n 比特的串, 对每个 $r \in \{0, 1\}^n$, 其中包含了 $b(x, r)$ 。具体而言, 函数 $B: \{0, 1\}^n \rightarrow \{0, 1\}$ 可以看作是长为 2^n 的串, 每个满足式 (7.7) 的 $x \in \{0, 1\}^n$ 对应于一个码字 (即对 x 的 Hadamard 编码), 且与 B 的距离至多为 $(0.5 - \varepsilon) \cdot 2^n$ 。定理 7.8 称, 当给定对 B 中比特的直接访问 (但是不能读出全部的 B) 时, 所有这些 x 的列表可在时间 $\text{poly}(n/\varepsilon)$ 内 (概率地) 恢复。这就显

① 关于 x 的可用信息在习题 7.14 中用 X_n 表示, 其中 x 本身用 $h(X_n)$ 表示。

② 事实上, s 的分布是任意的 (如习题 7.14 中的 X_n)。注意: 定理 7.7 可以看作是令 $\Pi(s)$ 在 $f^{-1}(s)$ 中均匀分布时的特例。

然得到了 Hadamard 编码的一种查表译码方法，其中查表译码的任务是恢复其码字与给定串距离在一定范围内的所有串，这不同于标准译码，后者的目标是恢复与给定串距离最近的码字中包含的唯一信息。我们指出，当在规定距离范围无法唯一译码时（即规定距离超过了码的一半距离时），查表译码还是有实用价值的（注意：在定理 7.7 证明一开始的讨论中，蕴含了一个 Hadamard 码的快速唯一译码程序）。

困难核心谓词的用途

现在回到困难核心谓词，我们认为它们在一般用途的伪随机数发生器（见 8.2 节）、承诺方案、零知识证明方案（见 9.2.2 节及 C.4.3）及加密方案（见附录 C.5）的构造中起着关键作用。

7.1.4 对困难放大的思考

现在注意定理 7.5 和定理 7.7 的本质。其中不仅仅是利用一个假设来得到某些结论；这是数学和自然科学中的常见练习（事实上，从定理 2.28 开始，前面的章节中已多次实践过这种练习）。定理 7.5 和定理 7.7 的特别之处在于（在 7.2 节、8.2 节及 8.3 节将进一步体现），证实了一个相对温和的困难性假设中蕴含着一个更强的困难性结论。

当人们承认自己不理解某种困难现象（因为不理解高效计算的本质）时，这种对困难的强化（即困难放大）就会起作用。不过，困难放大也可利用反证法实现，此时称为可约性方法。这看上去不那么令人激动：可约性方法要求设计一个程序（即一个归约）并对其行为进行概率分析。这种程序的构造及分析可能并不容易，但肯定会使用计算机科学的标准技术。定理 7.8 很好地体现了这个事实。

7.2 E 中的困难问题

与 7.1 节一样，从 $P \neq NP$ 的假设开始。本节将再次利用一个似乎更强的假设，这里，强化的意思是关于非一致的计算模型，求最坏情况下的困难性（而不是在标准的一致性计算模型下，求平均情况困难性）。具体地，我们将假设 NP 不能用多项式规模（非一致的）电路类求解，即 NP 类不包含于 $P/poly$ 中。

我们的目标是将这种最坏情况下的假设转化为平均情况条件，这对于要讨论的应用而言很有价值。由于转化之后得到的问题不属于 NP 而是属于 ε ，也可利用似乎更弱的假设，即 ε 不包含于 $P/poly$ 中（见习题 7.9）。也就是说，出发点实际上是存在一个指数时间可解的判定问题，除了有限多种输入长度之外，任意多项式规模的电路类无法对其求解。^①

从另一个角度看，这个假设是指 ε 中包含一些无法在多项式时间内求解的问题（见 4.2.1 节）。当前假设超出了这个事实，其中使用非一致的多项式时间机器，而不是（一致的）多项式时间机器。

回顾我们的目标是构造一个谓词（即一个判定问题），可在指数时间内计算，但是无法用多项式规模电路求近似。为了方便后续讨论，以下给出不可近似性的一般定义。

① 注意：这里的出发点实际上比假设 $\varepsilon \not\subseteq P/poly$ 中函数 f 的存在性更强。该假设意味着任意多项式规模电路类都无法对无穷多种长度的输入正确计算 f ，而我们假设电路类对除有限多种长度的输入之外都无法求值。

定义 7.9 (不可近似性的一般定义) 称函数 $f:\{0,1\}^* \rightarrow \{0,1\}$ 是 (S, ρ) -不可近似的, 如果对任意 S -规模的电路簇 $\{C_n\}_{n \in \mathbb{N}}$ 及所有足够大的 n , 有

$$\Pr[C_n(U_n) \neq f(U_n)] \geq \frac{\rho(n)}{2} \quad (7.8)$$

如果 f 是 $(T, 1-(1/T))$ -不可近似的, 则称其是 T -不可近似的。

我们选择式 (7.7) 的特殊形式, 使得由参数 ρ 所代表的“不可近似程度”在 $(0, 1)$ 范围内变化, 并随着 ρ 的增加而增大。具体地, 规模为 S 的电路在 (几乎处处) 最坏情况下的困难性可表示为 (S, ρ) -不可近似性, 其中 $\rho(n) = 2^{-n+1}$ (即此时对每个规模为 $S(n)$ 的电路 C_n , 有 $\Pr[C_n(U_n) \neq f(U_n)] \geq 2^{-n}$)。另一方面, 当 $\rho(n) = 1 - 2^{-n}$ 时, 不存在任何谓词是 (S, ρ) -不可近似的, 即使 $S(n) = O(n)$ (即对某些线性规模电路, 有 $\Pr[C_n(U_n) = f(U_n)] \geq 0.5 + 2^{-n-1}$, 见习题 7.10)。

注意: 式 (7.8) 可以理解为 f 的每个电路的相关性上界 (即式 (7.8) 等价于 $E[\chi(C(U_n), f(U_n))] \leq 1 - \rho(n)$, 其中当 $\sigma = \tau$ 时, $\chi(\sigma, \tau) = 1$, 否则, $\chi(\sigma, \tau) = -1$)。^①因此, T -不可近似性意味着不存在规模为 T 的电路类, 与 f 的相关性大于 $1/T$ 。

注意: 非一致的困难单向函数 (定义 7.3) 的存在性蕴含着存在指数时间可计算的谓词, 对所有多项式 T 是 T -不可近似的 (细节见习题 7.24)。然而, 本节的目标是在一个似乎更弱的假设下证明该结论。

几乎处处的困难性

这里要强调一个事实, 即所有相关性的假设和结论都针对着几乎处处的困难性。例如, 我们的出发点不仅仅是 ε 包含于 P/poly 中 (或其他要讨论的电路类中), 而是这在几乎处处都成立。注意: 这里所说的 f 的电路复杂度超过 S , 是指有无穷多个 n 使得不存在规模为 $S(n)$ 的电路可以对所有长为 n 的输入正确计算 f 。相反, 称 f 的电路复杂度几乎处处超过 S , 意思是除有限多个 n 外, 不存在规模为 $S(n)$ 的电路可对所有长为 n 的输入正确计算 f (事实上, 尚且不知道是否一个“无限几率”类型的困难性暗示着相应的“几乎处处”的困难性)。

ε 类

回顾 ε 表示指数时间可解的判定问题类 (或等价地, 指数时间可计算的布尔谓词), 即 $\varepsilon = \bigcup_{\varepsilon} D_{\text{TIME}}(t_{\varepsilon})$, 其中 $t_{\varepsilon}(n) \stackrel{\text{def}}{=} 2^{\varepsilon n}$ 。

本节的剩余内容

首先介绍关于多项式规模电路的假设及困难放大 (7.2.1 节), 这足以对 BPP 类进行非凡的去随机化。然后转向指数规模电路的假设及困难放大 (7.2.2 节), 从中得到对 BPP 类“完全的”去随机化 (即 $BPP = P$)。事实上, 这两个小节都包含一些可用于各种不同规模电路的内容, 但其动机是相同的。

教学注释: 7.2.2 节属于前沿性内容, 最好留作独立阅读。另外, 对其中的一个核心结论 (即引理 7.23), 只提供了概述, 有兴趣的读者可以参考原始论文^[128]。

^① 事实上, $E[\chi(X, Y)] = \Pr[X = Y] - \Pr[X \neq Y] = 1 - 2\Pr[X \neq Y]$ 。

7.2.1 对多项式规模电路的放大

本小节的目标是证明如下结论。

定理 7.10 假设对任意多项式 p , ε 中存在一个问题, 其电路复杂度几乎处处大于 p , 则 ε 中存在多项式-不可近似的布尔函数, 即对任意多项式 p , ε 中存在 p -不可近似的布尔函数。

定理 7.10 的用途是在上述假设下实现对 **BPP** 类有意义的去随机化 (见定理 8.19 的第 2 部分)。以下给出定理 7.10 的两种证明方法, 第一种方法由两个步骤构成。

(1) 从最坏情况下的假设开始, 首先证明关于平均情况困难性一些较温和的结论 (即较温和的不可近似性)。具体地, 证明对任意多项式 p , ε 中存在一个 (p, ε) -不可近似问题, 其中 $\varepsilon(n) = 1/n^3$ 。

(2) 利用上述较温和的不可近似性, 得到期望的强不可近似性 (即 p' -不可近似性, p' 为任意多项式)。具体地, 对任意两个多项式 p_1 和 p_2 , 证明如果函数 f 是 $(p_1, 1/p_2)$ -不可近似的, 则函数 $F(x_1, x_2, \dots, x_{t(n)}) = \bigoplus_{i=1}^{t(n)} f(x_i)$ 是 p' -不可近似的, 其中 $t(n) = n \cdot p_2(n)$, $x_1, x_2, \dots, x_{t(n)} \in \{0, 1\}^n$, $p'(t(n) \cdot n) = p_1(n)^{\Omega(1)} / \text{poly}(t(n))$ 。这就是所谓的 Yao 异或引理, 其证明远比信息论中类似结论的证明复杂得多 (7.2.1.2 节一开始讨论过)。

定理 7.10 的第二种证明中, 要证明将第一步的构造与定理 7.8 相结合, 可以得到期望的最终结论。这种方法将揭示困难放大与编码理论的联系。我们分 3 个步骤描述证明过程 (见 7.2.1.1 节~7.2.1.3 节, 图 7.2 是其示意图)。



图 7.2 困难放大的证明：组织结构

7.2.1.1 从最坏情况困难性到温和的平均情况困难性

将最坏情况困难性转化为平均情况困难性 (即使是在温和意义上) 是很诱人的。注意: 最坏情况困难性可能归结到少量实例, 而温和的平均情况困难性则针对着大量的可能实例。^① 换言之, 应将少数实例的困难性转化为大量实例的困难性。直观上, 应将 (原始问题的) 少数实例的困难性 “扩散” 到所有 (或大多数) (转化后的问题) 的实例。反过来, 如果能对转化后问题的典型实例求解, 则也能对原始问题的所有实例求解。

上述转化过程基于自校正 (Self-correction) 范式。该范式涉及到一个函数 g , 可利用 g 在几个随机点的值来对任意点求值, 这几个随机点在函数定义域中均匀分布 (但其实并不是

^① 事实上, 多项式规模电路在最坏情况下的困难性不能归结到多项式数量的实例, 因为多项式数量的实例可以被硬嵌入到这种电路中。然而, 众所周知, 最坏情况困难性可能来自于较小的超多项式数量的实例 (如 $n^{\log_2 n}$ 个实例)。相反, 即使是温和形式的平均情况困难性也一定来自于指数数量的实例 (如 $2^n / \text{poly}(n)$ 个实例)。

独立的)。关键在于, 如果可以利用 g 在 t 个随机点的值来重构 $g(x)$, 则这种重构可以容忍 $1/3t$ (关于 g 的值) 的错误。因此, 如果至少能得到 g 定义域内 $2/3t$ 的正确函数值, 则可以以极高的概率重构 g 在任意点的值。从而, 如果不存在最坏情况下正确计算 g 的概率多项式时间算法, 则任意概率多项式时间算法都无法对其定义域内 $1/3t$ 的点正确求值。

自校正函数一个典型的例子是有限域 F 上任意一个次数为 d 的 m 变量多项式, 满足 $|F| > dm+1$ 。该多项式在任意点 x 的值可利用 $dm+1$ 个点 (非 x) 的值求得, 这些点位于经过 x 的一条随机直线上。注意: 这些点中每一个都在 F^m 中均匀分布, 而 F^m 为多项式的定义域。(细节见习题 7.11)。

给定任意一个具有最坏情况困难性的函数 $f \in \varepsilon$, 该函数不一定是自校正的 (基于少数几个点), 但可以被扩展为自校正函数。具体地, 将函数 $f: [N] \rightarrow \{0,1\}$ (被看作是 $f: [N^{1/m}]^m \rightarrow \{0,1\}$) 扩展为有限域 F 上一个 d 次 m 变量多项式, 满足 $|F| > dm+1$ 且 $(d+1)^m = N$ 。直观上, 由最坏情况复杂度看, 扩展后的函数至少与 f 一样难, 这意味着它是困难的 (在最坏情况下)。由扩展后函数的自校正特性, 其最坏情况困难性蕴含了平均情况困难性。具体细节如下。

构造 7.11 (多变量扩展) ^①对任意函数 $f_n: \{0,1\}^n \rightarrow \{0,1\}$, 有限域 F , 集合 $H \subset F$, 以及整数 m , 满足 $|H|^m = 2^n$ 及 $|F| > (|H|-1)m+1$, 考虑对函数 $f_n: H^m \rightarrow \{0,1\}$ 扩展后的函数 $\hat{f}_n: F^m \rightarrow F$, 定义为 $|H|-1$ 次 m 变量多项式。也就是说, 将 $\{0,1\}^n$ 写成 H^m , 并定义 $\hat{f}_n$ 为唯一的 $|H|-1$ 次 m -变量多项式, 满足对任意 $x \in H^m$, $\hat{f}_n(x) = f_n(x)$, 这里将 $\{0,1\}$ 视为 F 的一个子集。

注意: 通过在整个定义域中计算 f_n , 并唯一确定与 f_n 在 H^m 上一致的 $|H|-1$ 次 m -变量多项式 (见习题 7.12), 可对 $\hat{f}_n$ 在任意点求值。因此, 对于 ε 中的函数 $f: \{0,1\}^* \rightarrow \{0,1\}$, 相应的 $\hat{f}$ (定义为对限定于每个输入长度的 f 进行独立扩展) 也属于 ε 。为使各种复杂性度量保持不变, 我们希望 $|F^m| = \text{poly}(2^n)$, 这就必须令 $m = n/\log_2 n$ (从而 $|H| = n$ 且 $|F| = \text{poly}(n)$)。特别地, 此时, $\hat{f}_n$ 定义在长为 $O(n)$ 的串上, 从而由上述讨论可得到 $\hat{f}$ 较温和的平均情况困难性。以下给出并证明一个更通用的结论。

定理 7.12 假设存在 ε 上的布尔函数 f , 其电路复杂性几乎处处大于 S , 则存在一个指数时间可计算的函数 $\hat{f}: \{0,1\}^* \rightarrow \{0,1\}^*$, 使得 $|\hat{f}(x)| \leq |x|$, 并对每个规模为 $S'(n') = S(n'/O(1))/\text{poly}(n')$ 的电路簇 $\{C'_{n'}\}_{n' \in \mathbb{N}}$, 有 $\Pr[C'_{n'}(U_{n'}) \neq \hat{f}(U_{n'})] > (1/n')^2$ 。进一步地, $\hat{f}$ 不依赖于 S 。

定理 7.12 似乎完成了定理 7.10 证明的第一步, 但我们期望的是得到一个布尔函数, 而不仅仅是无扩展的函数。为得到一个 $(\text{poly}(n), n^{-3})$ -不可近似的布尔函数, 还需采用与习题 7.13

① 这个构造所依据的代数知识是, 对任意函数 $f: H^m \rightarrow F$, 存在唯一的 $|H|-1$ 次 m -变量多项式 $\hat{f}: F^m \rightarrow F$, 使得对任意 $x \in H^m$, 有 $\hat{f}(x) = f(x)$ 。这个多项式称为 f 的多变量扩展, 它可在 $\text{poly}(|H|^m \log |F|)$ 时间内找到。具体细节见习题 7.12。

相同的步骤^①。实质上, 如果 $\hat{f}$ 对于占相当比例的输入都是困难的, 则对于输入 (x, i) , 能返回 $\hat{f}(x)$ 的第 i 比特的布尔谓词一定是温和不可近似的。

证明概要: 给定如假设中的 f , 对任意 $n \in \mathbb{N}$, 考虑将 f 限定于 $\{0, 1\}^n$ 上, 记作 f_n , 并对其应用构造 7.11, 同时设 $m = n / \log n$, $|H| = n$ 及 $n^2 < |F| = \text{poly}(n)$ 。此时得到的函数 $\hat{f}_n$ 将长为 $n' = \log_2 |F^m| = O(n)$ 的串映射到长为 $\log_2 |F| = O(\log n)$ 的串。根据上述讨论, 我们将证明, 从一个对 $\hat{f}_n$ 求近似的电路出发, 可以得到对每个输入正确计算 f_n 的电路。利用后者在规模上的假设, 可以得到前者的规模下限。实际(可约性)论证以如下方式进行。给定任意一个满足

$$\Pr[C'_{n'}(U_{n'}) = \hat{f}_n(U_{n'})] \geq 1 - (1/n')^2 > 1 - (1/3t) \quad (7.9)$$

的电路 $C'_{n'}$, 其中 $t \stackrel{\text{def}}{=} (|H| - 1)m + 1 = o(n^2)$ 大于 $\hat{f}_n$ 的整体次数。利用 $\hat{f}_n$ 的自校正性, 观察到通过向 $C'_{n'}$ 发出 t 个预言调用, 可以以至少 $2/3$ 的概率成功恢复 f_n 对每个输入的取值(从而恢复 $\hat{f}_n$ 对每个输入的取值)。利用错误缩减及定理 6.3 证明中的(非一致)去随机化^②, 可以得到一个规模为 $n^3 \cdot |C'_{n'}|$ 的计算 f_n 的电路。由假设 $n^3 \cdot |C'_{n'}| > S(n)$ 可知, $|C'_{n'}| > S(n'/O(1))/\text{poly}(n')$ 。注意到 $C'_{n'}$ 满足式 (7.9), 定理得证。□

思考

定理 7.12 的证明实际上是从最坏情况到平均情况的归约。证明中包含一个自校正程序, 能在任意期望的 n -比特点计算 f , 计算中需调用一个能在所有 n' -比特输入中, 对占 $1 - (1/n')^2$ 比例的点正确计算 $\hat{f}$ 的电路。回顾前面提到, 如果 $f \in \mathcal{E}$, 则 $\hat{f} \in \mathcal{E}$, 但是我们尚不知道当 f 属于 $\mathcal{NP}$ 时, 如何保持其复杂度(已知对于 $\mathcal{NP}$, 有各种最坏情况困难性到平均情况困难性的归约, 如参考文献[43])。

定理 7.12 的证明思想可用于各种环境。如在 9.3.2.1 节和 9.3.2.2 节中, 将应用自校正范式。另外, 在 9.3.2.2 节中, 还会遇到如定理 7.12 证明中所述的多变量扩展。

7.2.1.2 Yao 的异或引理

在得到了温和不可近似的谓词之后, 能否得到一个强不可近似的谓词呢? 在信息论背景下, 有一个吸引人的结论: 假设 X 是一个布尔变量(表示上述的温和不可近似谓词), 取值为 1 的概率是 ε , 则对 X 的 n/ε 个独立采样求异或, 得到结果为 1 的概率是 $0.5 \pm \exp(-\Omega(n))$ 。在计算理论背景下人们会期望同样的结论也成立, 即如果 f 是难求近似的, 这一事件成立的概率大于 $1 - \varepsilon$, 则对 f 的输入中占 n/ε 比例部分的输出求异或, 可以得到一个谓词, 对其求近似困难的概率将超过 $1/2$ 。人们发现后一个断言是正确的, 但(超过了 7.1.2 节)在计算理论中证明这一现象远比信息论中类似结论的证明复杂。

① 注意: 定理 7.12 的证明实际上证明了一个错误下限为 $\Omega(\log n')/(n')^2$, 根据这一点以及 $|\hat{f}(x)| = O(\log |x|)$, 可以得到一个在量级上更强的界限。

② 首先, 调用上述的概率算法 $O(n)$ 次, 并进行多数表决。得到一个概率算法, 对于输入 $x \in \{0, 1\}^n$, $o(n^3)$ 次调用 $C'_{n'}$, 能以大于 $1 - 2^{-n}$ 的概率正确计算 $f_n(x)$ 。最后, 可以固定一个随机序列, 对 2^n 种可能的输入都是良好的, 并得到一个规模为 $n^3 \cdot |C'_{n'}|$ 的电路, 能对所有 n 比特输入正确计算 f_n 。

定理 7.13 (Yao 的异或引理) 存在一个通用常数 $c > 0$, 使得以下结论成立。如果对两个多项式 p_1 、 p_2 , 布尔函数 f 是 $(p_1, 1/p_2)$ -不可近似的, 则函数 $F(x_1, x_2, \dots, x_{t(n)}) = \bigoplus_{i=1}^{t(n)} f(x_i)$ 是 p' -不可近似的, 其中 $t(n) = n \cdot p_2(n)$, $x_1, x_2, \dots, x_{t(n)} \in \{0, 1\}^n$, 而 $p'(t(n) \cdot n) = p_1(n)^c / t(n)^{1/c}$ 。进一步地, 如果将多项式 p_1 和 p_2 用任意整数函数代替, 断言仍成立。

结合定理 7.12 (及习题 7.13) 与定理 7.13, 可以得到定理 7.10 的一种证明 (回顾在 7.2.1.3 节中给出了另一种证明。)

注意: 证明定理 7.13 似乎比证明定理 7.5 (即对单向函数的放大) 更困难, 这有两个原因。首先, 与定理 7.5 不同, 计算问题不属于 $\mathcal{PC}$, 从而无法高效识别正确的解。其次, 不同于定理 7.5, 转化得到的实例的解并不是原始实例解的简单级联, 而是后者的一个函数, 并几乎丢失了关于后者的所有信息。以下对定理 7.13 的证明将分别解决这两个问题。

已知定理 7.13 有几种不同的证明方法。我们选择的方法在概念上更吸引人, 因为它分别独立地处理了两个不相关的难题。进一步地, 该证明可以利用 7.1 节的内容。其中包含以下两个步骤。

(1) 首先证明相应的“直积”函数 $P(x_1, x_2, \dots, x_{t(n)}) = (f(x_1), f(x_2), \dots, f(x_{t(n)}))$ 在较强的平均情况意义下是难计算的。

(2) 再利用定理 7.8 证明定理 7.13。

因此, 在已知定理 7.8 时, 证明的主要任务是第一步, 它本身也有独立的意义 (并可以从布尔函数推广到任意函数)。

定理 7.14 (直积引理) 令 p_1 、 p_2 为多项式, 并假设函数 $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 满足对任意 p_1 -规模电路簇 $\{C_n\}_{n \in \mathbb{N}}$ 及所有足够大的 $n \in \mathbb{N}$, 有 $\Pr[C_n(U_n) \neq f(U_n)] > 1/p_2(n)$ 。令 $P(x_1, x_2, \dots, x_{t(n)}) = (f(x_1), f(x_2), \dots, f(x_{t(n)}))$, 其中 $x_1, x_2, \dots, x_{t(n)} \in \{0, 1\}^n$, 且 $t(n) = n \cdot p_2(n)$ 。对任意 $\varepsilon': \mathbb{N} \rightarrow (0, 1]$, 取 p' 满足 $p'(t(n) \cdot n) = p_1(n) / \text{poly}(t(n) / \varepsilon'(t(n) \cdot n))$, 从而任意 p' -规模的电路类 $\{C'_m\}_{m \in \mathbb{N}}$ 均满足 $\Pr[C'_m(U_m) = P(U_m)] < \varepsilon'(m)$ 。进一步地, 如果将多项式 p_1 、 p_2 用任意整数函数代替, 上述结论仍成立。

特别地, 对于足够大的常数 $c > 0$, 选择 $\varepsilon'(t(n) \cdot n) = p_1(n)^{-c}$, 得到 $p'(t(n) \cdot n) = p_1(n)^c / t(n)^{1/c}$, 从而 $\varepsilon'(m) \leq 1/p'(m)$ 。

由定理 7.14 得到定理 7.13

定理 7.13 可由定理 7.14 推出, 考虑函数 $P'(x_1, x_2, \dots, x_{t(n)}, r) = b(f(x_1)f(x_2)\cdots f(x_{t(n)}), r)$, 其中 f 为布尔函数, $r \in \{0, 1\}^{t(n)}$, $b(y, r)$ 为 $t(n)$ -比特串 y 与 r 的模 2 内积。注意: 对相应的 P , 有 $P'(x_1, x_2, \dots, x_{t(n)}, r) \equiv b(P(x_1, x_2, \dots, x_{t(n)}), r)$, 而 $F(x_1, x_2, \dots, x_{t(n)}) = P'(x_1, x_2, \dots, x_{t(n)}, 1^{t(n)})$ 。直观上, P' 的不可近似性可由 P 的强平均情况困难性得到 (由定理 7.8), 而将对 P' 求近似归约到对 F 求近似也是可能的 (从而得到 F 的不可近似性)。事实上, 这个直觉是正确的, 但实现细节十分麻烦 (这里只提供了思路)。假设 f 是 $(p_1, 1/p_2)$ -不可近似的, 首先应用定理 7.14 (令 $\varepsilon'(t(n) \cdot n) = p_1(n)^{-c}$), 再应用定理 7.8 (见习题 7.14), 得到 P' 的 p' -不可近似性, 其中 $p'(t(n) \cdot n) = p_1(n)^{c(1)} / \text{poly}(t(n))$ 。这里有难度的部分是将 P' 的近似归约为对 F 求近似。关

键在于

$$P'(x_1, x_2, \dots, x_{l(n)}, r) = F(z_1, z_2, \dots, z_{l(n)}) \oplus_{i:r_i=0} f(z_i) \quad (7.10)$$

其中当 $r_i = 1$ 时, $z_i = x_i$, 否则, z_i 为任意的 n 比特串。现在, 如果得到了分布 $(U_n, f(U_n))$ 的样本, 则可利用这些样本作为 $(z_i, f(z_i))$, 其中 i 满足 $r_i = 0$ 。从这些样本的一种最佳选择 (即从中能得到 P' 的最佳近似) 出发, 可以得到一个对 P' 求近似的电路 (利用对 F 求近似的电路以及该样本)。(细节留作习题 7.17) 定理 7.13 得证。

定理 7.14 的证明

注意: 定理 7.14 与定理 7.5 密切相关, 细节见习题 7.20。这表明, 可以利用与定理 7.5 类似的证明策略, 将不满足定理结论的电路转化为不满足假设的电路。然而, 要注意, 定理 7.5 是一种简单情况: 可以高效地检查包含于解直积问题电路的输出中的值是否为基本问题对应实例的正确解。定理 7.14 没有这么简单, 必须更谨慎地使用这些值。不严格地说, 应该在各种应答中采取加权的多数表决, 其中权值体现了对各种应答正确性的确信程度。

证明定理 7.14 时, 可以利用如下引理, 其中给出了计算两个函数直积可行性的量化上限。在该引理中, $\{Y_m\}_{m \in \mathbb{N}}$ 和 $\{Z_m\}_{m \in \mathbb{N}}$ 为独立的概率空间, 满足 $Y_m, Z_m \in \{0, 1\}^m$, 且对某个函数 $l: \mathbb{N} \rightarrow \mathbb{N}$, 有 $X_n = (Y_{l(n)}, Z_{n-l(n)})$ 。引理涉及到计算由式 $F(yz) = (F_1(y), F_2(z))$ 所定义的直积函数 $F: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 的成功概率, 其中 $|y| = l(|yz|)$, 且给定了 (独立) 计算 F_1 和 F_2 的成功概率上限。当然, 这些概率上限只针对具有特定规模的电路 (松弛参数 ε 表示与理想结果的偏移程度, 在理想结果中, 正确计算 F 的概率上限是正确计算 F_1 和 F_2 的概率之积, 而要使这个参数更加紧凑 (即减小 ε), 需减小满足成功概率上限的电路规模)。我们强调该引理关于两个函数不是对称的: 它保证对其中一个函数 (这里任意地将其定为 F_1) 能更好地保持 (实际上不变) 电路规模。

引理 7.15 (直积, 量化的二元版本) 对于上述的 $\{Y_m\}$ 、 $\{Z_m\}$ 、 F_1 、 F_2 、 l 、 $\{X_n\}$ 及 F , 令 $\rho_1(\cdot)$ 为在 $\{Y_m\}$ 上, 规模为 $s_1(\cdot)$ 的电路成功计算 F_1 的概率上限。也就是说, 对任意这样的电路簇 $\{C_m\}$, 有

$$\Pr[C_m(Y_m) = F_1(Y_m)] \leq \rho_1(m)$$

同样, 假设 $\rho_2(\cdot)$ 为 $\{Z_m\}$ 上, 规模为 $s_2(\cdot)$ 的电路成功计算 F_2 的概率上限, 则对任意函数 $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}$, 下式定义的函数 ρ 为

$$\rho(n) \stackrel{\text{def}}{=} \rho_1(l(n)) \cdot \rho_2(n-l(n)) + \varepsilon(n)$$

是 $\{X_n\}$ 上, 规模为 $s(\cdot)$ 的电路成功计算 F 的概率上限, 其中

$$s(n) \stackrel{\text{def}}{=} \min \left\{ s_1(l(n)), \frac{s_2(n-l(n))}{\text{poly}(n/\varepsilon(n))} \right\}$$

定理 7.14 可利用归纳法由引理 7.15 得出, 其中使用引理 7.15 的量化形式, 特别是依赖于一个事实, 即所使用的两个函数之一是无损失的。首先描述这个讨论的细节, 再证明引理 7.15。

由引理 7.15 推出定理 7.14

将 $P(x_1, x_2, \dots, x_{l(n)})$ 写成 $P^{(l(n))}(x_1, x_2, \dots, x_{l(n)})$, 其中 $P^{(i)}(x_1, x_2, \dots, x_i) = (f(x_1), f(x_2), \dots,$

$f(x_i)$ 且 $P^{(i)}(x_1, x_2, \dots, x_i) \equiv (P^{(i-1)}(x_1, x_2, \dots, x_{i-1}))$ 。给定任意函数 ε ，对 i 进行归纳，我们证明规模为 $s(n) = p_1(n)/\text{poly}(t(n)/\varepsilon(n))$ 的电路不能以大于 $(1 - (1/p_2(n)))^i + (i-1) \cdot \varepsilon(n)$ 的概率成功计算 $P^{(i)}(U_{i,n})$ ，其中 p_1 和 p_2 如定理 7.14 所示。因此，不存在 $s(n)$ -规模的电路能以大于 $(1 - (1/p_2(n)))^{t(n)} + (t(n)-1) \cdot \varepsilon(n) = \exp(-n) + (t(n)-1)$ 的成功概率计算 $P^{(t(n))}(U_{t(n),n})$ 。可对任意函数 ε 证明这一结论成立，从而定理 7.14 得证（利用 $\varepsilon(n) = \varepsilon'(t(n) \cdot n)/t(n)$ ，并观察到令 $s(n) = p'(t(n) \cdot n)$ 即可满足 $s(n) = p_1(n)/\text{poly}(t(n)/\varepsilon(n))$ ）。

现在转向归纳本身，首先要注意归纳的基础步骤（即 $i=1$ 情形）可由定理假设保证（即定理 7.14 中关于 f 的假设）。为了证明归纳步骤（由 i 到 $i+1$ ），可利用引理 7.15，并设 $F_1 = P^{(i)}$ 而 $F_2 = f$ ，同时设置参数为 $\rho_1^{(i)}(i \cdot n) = (1 - (1/p_2(n)))^i + (i-1) \cdot \varepsilon(n)$ ， $s_1^{(i)}(i \cdot n) = s(n)$ ， $\rho_2^{(i)}(n) = (1 - (1/p_2(n)))$ 以及 $s_2^{(i)}(n) = \text{poly}(n/\varepsilon(n)) \cdot s(n) = p_1(n)$ 。具体细节如下。

注意到，（关于 $P^{(i)}$ 的）归纳假设暗示了 F_1 满足引理 7.15 的假设（关于规模 $s_1^{(i)}$ 及成功率 $\rho_1^{(i)}$ ），而定理中关于 f 的假设暗示了 F_2 满足引理 7.15 的假设（关于规模 $s_2^{(i)}$ 及成功率 $\rho_2^{(i)}$ ）。因此， $F = P^{(i+1)}$ 满足引理中关于电路规模 $\min(s_1^{(i)}(i \cdot n), s_2^{(i)}(n)/\text{poly}(n/\varepsilon(n))) = s(n)$ 及成功率 $\rho_1^{(i)}(i \cdot n) \cdot \rho_2^{(i)}(n) + \varepsilon(n)$ 的结论，其成功率上限为 $(1 - (1/p_2(n)))^{i+1} + i \cdot \varepsilon(n)$ 。这就完成了归纳步骤。

我们强调一个事实，归纳的步骤数量并非常数，这是由于归纳断言的强量化形式以及归纳步骤导致的少量丢失。具体而言，规模上限在归纳中不会减小（虽然在每一步中可以容忍一定数量的减小，但减小的量并未达到一个常数因子）^①。类似地，在每一步，要求成功率增加 $\varepsilon(n)$ ，这也可由归纳断言来保证。因此，假设引理 7.15 正确，则可证明定理 7.14。□

引理 7.15 的证明

用反证法，考虑不符合引理结论的 $s(\cdot)$ -规模的电路簇 $\{C_n\}_{n \in \mathbb{N}}$ ，即 $\Pr[C_n(X_n) = F(X_n)] > \rho(n)$ 。我们证明如何利用这种电路来得到另一个电路，或者不满足引理中关于 F_1 的假设，或者不满足关于 F_2 的假设。为此，有必要将 C_n 的成功概率写成条件形式，而将 $C_n(x)$ 的第 i 个输出记作 $C_n(x)_i$ （即 $C_n(x) = (C_n(x)_1, C_n(x)_2)$ ）：

$$\begin{aligned} & \Pr[C_n(Y_{l(n)}, Z_{n-l(n)}) = F(Y_{l(n)}, Z_{n-l(n)})] \\ &= \Pr[C_n(Y_{l(n)}, Z_{n-l(n)})_1 = F_1(Y_{l(n)})] \cdot \Pr[C_n(Y_{l(n)}, Z_{n-l(n)})_2 = F_2(Z_{n-l(n)}) | C_n(Y_{l(n)}, Z_{n-l(n)})_1 = F_1(Y_{l(n)})] \end{aligned}$$

这里的基本思路是：如果第一个因子大于 $\rho_1(l(n))$ ，则可直接得到一个电路（即 $C'_n(y) = C_n(y, Z_{n-l(n)})_1$ ），与引理关于 F_1 的假设矛盾。如果第二个因子远远大于 $\rho_2(n-l(n))$ ，

① 注意：如果（在引理 7.15 中）令 $s(n) = \min\{s_1(l(n)), s_2(n-l(n))\}/2$ ，则由上述讨论可得出对于规模为 $s_1^{(i+1)}$ 的电路， $P^{(i+1)}$ 是困难的，其中电路规模满足 $s_1^{(i+1)}((i+1) \cdot n) = \min\{s_1^{(i)}(i \cdot n), s_2(n)\}/2$ ，而对于 $P = P^{(t(n))}$ ，这将得到无意义的结论（因为 $s_1^{(t(n))}(t(n) \cdot n) = \min\{p_1(n), p_2(n)\}/2^{t(n)}$ ，其中 $t(n) \gg n$ 且 $p_i(n) < 2^n$ ）。

则可得到一个电路, 与引理关于 F_2 的假设矛盾。后一种情况不太容易处理。思路是可将一个足够大的样本 $(Y_{l(n)}, F_1(Y_{l(n)}))$ 硬嵌入到电路中, 从而在对 F_2 求近似时, 可以使用条件概率空间 (在该电路中)。也就是说, 对于输入 z , 均匀选择一个串 y , 满足 $C_n(y, z)_1 = F_1(y)$ (在上述样本中), 并输出 $C_n(y, z)_2$ 。对于给定的 z , 条件空间中的采样 (即满足 $C_n(y, z)_1 = F_1(y)$ 的 y) 很可能使 $\Pr[C_n(Y_{l(n)}, z)_1 = F_1(Y_{l(n)})]$ 以显著概率成立。最后一个说明要求独立处理使 $\Pr[C_n(Y_{l(n)}, z)_1 = F_1(Y_{l(n)})]$ 的取值不可忽略的 z , 以及其余的 z (基本上可以忽略)。细节如下。

首先简化符号, 将 n 固定下来, 并使用缩写, $C = C_n$, $\varepsilon = \varepsilon(n)$, $l = l(n)$, $Y = Y_l$, 以及 $Z = Y_{n-l}$ 。如果 $\Pr[C(Y, z)_1 = F_1(Y)] \geq \varepsilon/2$, 则称 z 是良好的, 令 G 为所有良好 z 的集合。然后, 并不考虑事件 $C(Y, Z) = F(Y, Z)$, 而是讨论联合事件 $C(Y, Z) = F(Y, Z) \wedge Z \in G$, 这两者发生的概率几乎相同 (最多相差 $\varepsilon/2$)。因为对任意 $z \notin G$, 有

$$\Pr[C_n(Y, z) = F(Y, z)] \leq \Pr[C_n(Y, z)_1 = F_1(Y)] < \varepsilon/2$$

从而非良好的 z 对于概率 $\Pr[C(Y, Z) = F(Y, Z)]$ 并无太多贡献; 也就是说, $\Pr[C(Y, Z) = F(Y, Z) \wedge Z \in G]$ 的下限为 $\Pr[C(Y, Z) = F(Y, Z)] - \varepsilon/2$ 。利用 $\Pr[C_n(Y, z) = F(Y, z)] > \rho(n) = \rho_1(l) \cdot \rho_2(n-l) + \varepsilon$, 有

$$\Pr[C(Y, Z) = F(Y, Z) \wedge Z \in G] > \rho_1(l) \cdot \rho_2(n-l) + \frac{\varepsilon}{2} \quad (7.11)$$

根据上述概要继续证明。首先证明如果 $\Pr[C(Y, Z)_1 = F_1(Y)] > \rho_1(l)$, 则直接可得到不满足关于 F_1 假设的电路。实际上, 可以证明一个更强的结论 (在其他情况中会需要)。

断言 7.15.1 对任意 z , 有 $\Pr[C(Y, Z)_1 = F_1(Y)] \leq \rho_1(l)$ 。

证明: 如果断言不成立, 利用任意满足 $\Pr[C(Y, Z)_1 = F_1(Y)] > \rho_1(l)$ 的 $z \in \{0, 1\}^{n-l}$, 可得到电路 $C'(y) \stackrel{\text{def}}{=} C(y, z)_1$, 与引理中关于 F_1 的假设矛盾。□

利用断言 7.15.1, 可以说明如何得到一个不符合引理中关于 F_2 假设的电路, 这就完成了引理的证明。

断言 7.15.2 存在规模为 $s_2(n-l)$ 的电路 C'' , 满足

$$\Pr[C'' = F_2(Z)] \geq \frac{\Pr[C(Y, Z) = F(Y, Z) \wedge Z \in G]}{\rho_1(l)} - \frac{\varepsilon}{2} \geq \rho_2(n-l)$$

证明: 第二个不等式可由式 (7.11) 得到, 所以只需证明第一个不等式。根据以上描述的要害构造电路 C'' 。具体地, 如果有了服从分布 $(Y, F_1(Y))$ 的 $\text{poly}(n/\varepsilon)$ 个采样, 记作 S , 则可令 $C''(z) \stackrel{\text{def}}{=} C(y, z)_2$, 其中 (y, v) 在 S 中均匀选择, 并满足 $C(y, z)_1 = v$ 。具体细节如下。

令 m 为足够大的整数, 上限是关于 n/ε 的多项式, 考虑随机选择的 m 个对, 生成方法是从分布 $(Y, F_1(Y))$ 中独立选择 m 个样本。我们强调, 这里并不假设这样一个样本, 记作 S , 可由一个高效的 (一致性) 算法产生 (但是指出这个序列可以固定下来)。对任意 $z \in G \subseteq \{0, 1\}^{n-l}$, 将满足 $C(y, z)_1 = v$ 的对 $(y, v) \in S$ 构成的集合记作 S_z 。注意: S_z 是由 $(Y, F_1(Y))$ 定义的剩余概

率空间中一个随机样本, 满足条件 $C(Y, z)_1 = F_1(Y)$ 。另外, $|S_z| = \Omega(n/\varepsilon^2)$ 以显著优势成立, 这是因为 $z \in G$ 蕴含了 $\Pr[C(Y, z)_1 = F_1(Y)] \geq \varepsilon/2$ 且 $m = \Omega(n/\varepsilon^3)$ 。^①因此, 对任意 $z \in G$, 样本 S_z 以显著优势 (在所有可选的 S 中) 提供对条件概率空间的良好近似。^②特别地, 以下不等式

$$\frac{|\{(y, v) \in S_z : C(y, z)_2 = F_2(z)\}|}{|S_z|} \geq \Pr[C(y, z)_2 = F_2(z) | C(y, z)_1 = F_1(z)] - \frac{\varepsilon}{2} \quad (7.12)$$

成立的概率大于 $1 - 2^{-n}$ 。因此, 对所有 $z \in G \subseteq \{0, 1\}^{n-1}$, 式 (7.12) 成立的概率为正值。计算 F_2 的电路 C'' 可以定义如下。电路中包含一个集合 $S = \{(y_i, v_i) : i = 1, 2, \dots, m\}$ (即 S 被“硬嵌入”到 C'' 中), 满足以下两个条件。

(1) 对任意 $i \in [m]$, 有 $v_i = F_1(y_i)$ 。

(2) 对任意良好的 z , 集合 $S_z = \{(y, v) \in S : C(y, z)_1 = v\}$ 满足式 (7.12)。

特别地, 对任意良好的 z , S_z 非空。

对于输入的 z , 电路 C'' 首先确定集合 S_z , 方法是将 C 运行 m 次, 并对每个 $i = 1, 2, \dots, m$, 检查是否有 $C(y_i, z) = v_i$ 。如果 S_z 为空, 则返回任意值; 否则, 均匀选择一对 $(y, v) \in S_z$, 并输出 $C(y, z)_2$ (这里的随机选择可用平均情况来代替, 见习题 7.16)。利用 C'' 的定义及式 (7.12), 有

$$\begin{aligned} \Pr[C''(Z) = F_2(Z)] &\geq \sum_{z \in G} \Pr[Z = z] \cdot \Pr[C''(z) = F_2(z)] \\ &= \sum_{z \in G} \Pr[Z = z] \cdot \frac{|\{(y, z) \in S_z : C(y, z)_2 = F_2(z)\}|}{|S_z|} \\ &\geq \sum_{z \in G} \Pr[Z = z] \cdot \left(\Pr[C(Y, z)_2 = F_2(z) | C(Y, z)_1 = F_1(Y)] - \frac{\varepsilon}{2} \right) \\ &= \sum_{z \in G} \Pr[Z = z] \cdot \left(\frac{\Pr[C(Y, z)_2 = F_2(z) \wedge C(Y, z)_1 = F_1(Y)]}{\Pr[C(Y, z)_1 = F_1(Y)]} - \frac{\varepsilon}{2} \right) \end{aligned}$$

然后, 利用断言 7.15.1, 有

$$\begin{aligned} \Pr[C''[Z] = F_2(Z)] &\geq \left(\sum_{x \in G} \Pr[Z = z] \cdot \frac{\Pr[C(y, z) = F(y, z)]}{\rho_1(l)} \right) - \frac{\varepsilon}{2} \\ &= \frac{\Pr[C(Y, Z) = F(Y, Z) \wedge Z \in G]}{\rho_1(l)} - \frac{\varepsilon}{2} \end{aligned}$$

最后利用断言 7.15.1, 断言得证。 □

这就完成了引理的证明。 ■

说明: 首先, 希望读者注意可约性方法在计算环境中执行时, 需要考虑的问题, 特别是

① 注意: S_z 的期望规模为 $m \cdot \varepsilon/2 = \Omega(n/\varepsilon^2)$ 。利用 Chernoff 界, 有 $\Pr_S[|S_z| < m\varepsilon/4] = \exp(-\Omega(n/\varepsilon^2)) < 2^{-n}$ 。

② 对于 $T_z = \{y : C(y, z)_1 = F_1(y)\}$, 我们感兴趣的是 T_z 的一个样本 S' , 满足 $|\{y \in S' : C(y, z)_2 = F_2(z)\}|/|S'|$ 近似等于 $\Pr[C(Y, z)_2 = F_2(Y) | Y \in T_z]$, 两者相差 $\varepsilon/2$ 。再次利用 Chernoff 界, 一个随机的规模为 $\Omega(n/\varepsilon^2)$ 的 $S' \in T_z$ 能以大于 $1 - 2^{-n}$ 的概率提供这种近似。

当需要的归纳步骤数不是一个常数时。事实上,定理 7.14 的归纳证明中调用了一个量化引理(即引理 7.15),可以在归纳过程中记录相关的量(如成功概率及电路规模)。其次,引理 7.15(以及定理 7.14)存在一个复杂性一致的版本,其中假设可以高效地从分布 $(Y_{l(n)}, F_1(Y_{l(n)}))$ (或 $(U_n, f(U_n))$) 中采样。细节见参考文献[102]。事实上,引理 7.15 证明最有价值之处在于可以利用非一致的电路对任意分布“高效采样”。最后,我们指出定理 7.5(单向函数的放大)及定理 7.13(Yao 的异或引理)也有(紧凑的)量化版本(分别见于参考文献[91]中 2.3.2 节和参考文献[102]中第 3 节)。

7.2.1.3 查表译码与困难放大

回顾 7.2.1.1 节和 7.2.1.2 节中证明了定理 7.10, 方法是根据构造 7.11 来构造一个温和不可近似谓词, 然后利用 Yao 的异或引理放大其困难性。本小节将证明第一步中使用的构造(即构造 7.11)实际上可得到一个强不可近似谓词, 从而给出定理 7.10 的另一种证明。具体地, 证明将构造 7.11(适当选择参数)与内积构造(定理 7.8)相结合, 可以得到一个强不可近似谓词(如定理 7.10 所述)。即如下结论。

命题 7.16 假设 ε 中存在布尔函数 f , 其电路复杂性处处大于 S , 并设 $\varepsilon: \mathbf{N} \rightarrow [0, 1]$ 满足 $\varepsilon(n) > 2^{-n}$ 。令 f_n 表示将 f 限定于 $\{0, 1\}^n$ 的函数, 并设 $\hat{f}_n$ 为对 f_n 应用构造 7.11^①, 并令其中 $|H| = n/\varepsilon(n)$ 和 $|F| = |H|^3$ 时, 所得到的函数。则由 $\hat{f}(x) = \hat{f}_{|x|/3}(x)$ 定义的函数 $\hat{f}: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 可在指数时间内计算, 且对任意规模为 $S'(n') = \text{poly}(\varepsilon(n'/3)/n') \cdot S(n'/3)$ 的电路簇 $\{C'_{n'}\}_{n' \in \mathbf{N}}$, 有 $\Pr[C'_{n'}(U_{n'}) = \hat{f}(U_{n'})] < \varepsilon'(n') \stackrel{\text{def}}{=} \varepsilon(n'/3)$ 。

在证明命题 7.16 之前, 先阐述如何从中得到定理 7.10 的另一种证明。对某个 $\gamma > 0$, 由命题 7.16 可得到一个指数时间计算的函数 $\hat{f}$, 满足 $|\hat{f}(x)| \leq |x|$, 且对任意规模为 $S'(n') = S(n'/3)^\gamma / \text{poly}(n')$ 的电路簇 $\{C'_{n'}\}_{n' \in \mathbf{N}}$, 有 $\Pr[C'_{n'}(U_{n'}) = \hat{f}(U_{n'})] < 1/S'(n')$ 。与定理 7.8 相结合(见习题 7.14), 可以推断出对于 $S''(n'') = S'(n''/2)^{\Omega(1)} \text{poly}(n'')$, $P(x, r) = b(\hat{f}(x), r)$ 是 S'' -不可近似的, 其中 $|r| = |\hat{f}(x)| \leq |x|$ 。特别地, 对任意多项式 p , 在上述讨论中令 $S(n) = \text{poly}(n, p(n))$, 可得到 ε 中一个 p -不可近似谓词。定理 7.10 得证。

教学注释: 以下内容最好留作独立阅读。另外, 要理解这些内容, 需要非常熟悉纠错码的相关概念(见附录 E.1.1)。

证明命题 7.16 时, 注意到由 f_n 向 $\hat{f}_n$ 的转化构成了一种“强”编码, 这种编码能提供从最坏情况到(强)平均情况的归约。初学者需注意, 映射 $f_n \mapsto \hat{f}_n$ 与 Reed-Muller 码(见附录 E.1.2.4)密切相关, 而我们已经看到了“困难放大”与编码理论之间的联系(见 7.1.3 节

① 回顾在构造 7.11 中, 有 $|H|^m = 2^n$, 如果要求 $|H| = n/\varepsilon(n)$, 则得到的 m 可能不是一个整数。定理 7.12 的证明中避免了这个问题(其中使用 $|H| = n$), 但本节由于 ε 的取值(如可能有 $\varepsilon(n) = 2^{-2n/7}$), 该问题将更为突出。因此, 或者应该放宽 $|H|^m = 2^n$ 的要求(如允许 $2^n \leq |H|^m < 2^{2n}$), 或者放宽 $|H| = n/\varepsilon(n)$ 的要求。然而, 为简单起见, 后面的阐述中忽略了这一点。

结尾的讨论)。在本小节环境中(将最坏情况下计算 f_n 归约为平均情况下计算 $\hat{f}_n$)，我们将寻找一个相对较小的电路来计算 f_n (对每个输入都能正确计算)，条件是给定到任意函数 f' 的预言访问，其中 f' 与 $\hat{f}_n$ 对于相当大一部分输入都是一致的。实际上，可以放宽这个要求(改变量词的顺序)，允许(有预言帮助的)小规模电路依赖于 f' (以及 f_n)，因为 f' 代表某个小规模电路，与关于 $\hat{f}_n$ 的平均情况复杂性要求不一致(从而组合后的电路将与关于 f_n 的最坏情况假设矛盾)。进一步地，为了让 $f_n \in \varepsilon$ 蕴含着 $\hat{f}_n \in \varepsilon$ ，我们希望映射 $f_n \mapsto \hat{f}_n$ 能在时间 $2^{O(n)} = \text{poly}(|f_n|)$ 内计算。基于这些考虑，定义如下的编码映射 $f_n \mapsto \hat{f}_n$ 。

定义 7.17 (支持绝对译码的高效编码) 对于给定函数 $q, l: \mathbf{N} \rightarrow \mathbf{N}$ 及 $\alpha: \mathbf{N} \rightarrow (0, 1]$ ，称映射 $\Gamma: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 是高效的且支持参数为 q, l, α 的绝对译码，如果它满足以下两个条件。

(1) 编码(或效率)。映射 Γ 可在多项式时间内计算。

可以认为 Γ 将 N -比特串映射为 $[q(N)]$ 上的 $l(N)$ 长序列，并将每个(码字) $\Gamma(x) \in [q(|x|)]^{l(|x|)}$ 看作是从 $[l(|x|)]$ 到 $[q(|x|)]$ 的映射。

(2) 译码(明确地)。存在一个多项式 p ，满足以下条件。对任意 $\omega: [l(N)] \rightarrow [q(N)]$ 及任意 $x \in \{0, 1\}^N$ ， $\Gamma(x)$ 与 ω 是 $(1 - \alpha(N))$ -接近的，从而存在一个有预言帮助的^①、规模为 $p((\log N)/\alpha(N))$ 的电路 C ，使得对任意 $i \in [N]$ ， $C^\omega(i)$ 等于 x 的第 i 比特。

编码条件蕴含了 l 的多项式上限。译码条件则说明：任意一个 Γ -码字与预言 $\omega: [l(N)] \rightarrow [q(N)]$ 在 $l(N)$ 个系数中有占 $\alpha(N)$ 比例的系数是一致的，其中 $\alpha(N)$ 可能很小。我们强调译码条件的非平凡性： x 中有 N 个比特，而电路 C 可能只对 x 中 $p((\log N)/\alpha(N))$ 比特的信息“编码”。因此， x 可以(明确地)由 C 恢复，主要根据 $\Gamma(x)$ 一个变化较大的版本。进一步地，在恢复 x 中每个比特时，最多只向 $\Gamma(x)$ 发出 $p((\log N)/\alpha(N))$ 个询问。上述译码条件与查表译码(定义见附录 E.1.1)相关。^②

现在将 f_n 到 $\hat{f}_n$ 的变换与定义 7.17 相关联，该变换是命题 7.16 的基础。将 f_n 看作是长为 $N = 2^n$ 的二元串(表示 $f_n: H^m \rightarrow \{0, 1\}$ 的真值表)，类似地，将 $\hat{f}_n: F^m \rightarrow F$ 看作是 $F^{|F|^m} = F^{N^3}$ 的一个元素(或从 $[N^3]$ 到 $[F]$ 的一个映射)。^③回想 f_n 到 $\hat{f}_n$ 的变换可以高效计算。该变换还支持参数为 q, l, α 的绝对译码，其中 $l(N) = N^3$ ， $\alpha(N) = \varepsilon(n)$ ， $q(N) = (n/\varepsilon(n))^3$ ， $N = 2^n$ 。

① 有预言帮助的电路与预言图灵机的定义类似。作为替代，在这里考虑接受建议的预言机，其中建议的长度及机器的运行时间上限均为 $p((\log N)/\alpha(N))$ 。相关的预言或被看作是对 $[q(N)]$ 上序列编码后的二元串，或被看作 $[q(N)]$ 上的序列。事实上，后一种情况考虑非二进制的预言，其返回的元素属于 $[q(N)]$ 。

② 回顾对于输入 $\omega \in [q(N)]^{l(N)}$ ， $\Gamma: \{0, 1\}^N \rightarrow [q(N)]^{l(N)}$ 的查表译码算法要求输出包含所有串 $x \in \{0, 1\}^N$ 的表 L_ω ，使得 $\Gamma(x)$ 与 ω 在 $\alpha(N)$ 比例的位置一致。现在转向上述的译码条件(定义 7.17)，注意到，其中要求输出某个特定的串 $x \in L_\omega$ ，为此，可将 x (在查表译码电路中)的索引硬嵌入到 L_ω 中。然而，要注意，用这种方法得到的电路可能不满足定义 7.17 中的严格译码条件，其中要求电路规模为 $p((\log N)/\alpha(N))$ 。另一方面，译码条件中没有提及为得到上述有预言帮助电路所用算法的复杂度(尤其是可能得不到一个查表译码算法)。

③ 回顾 $N = 2^n = |H|^m$ 且 $|F| = |H|^3$ ，因此 $|F|^m = N^3$ 。

这一事实绝非显然,但其证明超出了本章的讨论范围(感兴趣的读者可见参考文献[218])。

f_n 到 $\hat{f}_n$ 的变换还有其他一些性质,这些性质在定义 7.17 中并未要求,本章也不涉及。例如,至多有 $O(1/\alpha(2^n)^2)$ 个码字(即 $\hat{f}_n$)与任意给定的 $\omega: [l(2^n)] \rightarrow [q(2^n)]$ 是 $(1-\alpha(2^n))$ -接近的,并且相应的有预言帮助的电路可以在 $p(n/\alpha(2^n))$ -时间内概率地构造。^①这些结论被称为“具有明确表示的查表译码”(建议感兴趣的读者可见参考文献[218])。

我们的重点是要证明支持绝对译码的高效编码方案足以构成由(强)最坏情况向平均情况的归约。这里叙述并证明一个通用结论,注意:在命题 7.16 的特例中 $g_n = \hat{f}_n$ (且 $l(2^n) = 2^{3n}$)。

定理 7.18 假设 ε 中存在布尔函数 f , 其电路复杂度几乎处处大于 S , 令 $\varepsilon: \mathbf{N} \rightarrow (0,1]$ 。考虑多项式 $l: \mathbf{N} \leftarrow \mathbf{N}$, 满足 $n \mapsto \log_2 l(2^n)$ 是整数上的一一映射, 并令 $m(n) = \log_2 l(2^n)$, 如可令 $l(N) = N^3$, 则 $m(n) = 3n$ 。假设映射 $\Gamma: \{0,1\}^* \rightarrow \{0,1\}^*$ 可以高效计算, 并支持参数为 q, l, α 的绝对译码, 满足 $\alpha(N) = \varepsilon(\lfloor \log_2 N \rfloor)$ 。定义 $g_n: [l(2^n)] \rightarrow [1(2^n)]$, 使得 $g_n(i)$ 等于 $\Gamma(\langle f_n \rangle) \in [q(2^n)]^{l(2^n)}$ 的第 i 个元素, 其中 $\langle f_n \rangle$ 表示 f_n 真值表的 2^n 比特描述。由 $g(z) = g_{m^{-1}(\lfloor z \rfloor)}(z)$ 定义的函数 $g: \{0,1\}^* \rightarrow \{0,1\}^*$ 可在指数时间内计算, 且对任意规模为 $S'(n') = \text{poly}(\varepsilon(m^{-1}(n'))/n') \cdot S(m^{-1}(n'))$ 的电路簇 $\{C'_{n'}\}_{n' \in \mathbf{N}}$, 有 $\Pr[C'_{n'}(U_{n'}) = g(U_{n'})] < \varepsilon'(n') \stackrel{\text{def}}{=} \varepsilon(m^{-1}(n'))$ 。

证明概要: 首先注意到 f_n 的真值表可在指数时间内生成, 且由 Γ 的译码条件, g_n 可在指数时间内计算。 g 的平均情况困难性通过如下的可约性方法证明。考虑规模为 S' 的电路 $C' = C'_{n'}$, 满足 $\Pr[C'_{n'}(U_{n'}) = g(U_{n'})] < \varepsilon'(n')$, 令 $n = m^{-1}(n')$, 并注意 $\varepsilon'(n') = \varepsilon(n) = \alpha(2^n)$ 。于是, $C': \{0,1\}^{n'} \rightarrow \{0,1\}$ (视为一个函数)与函数 g_n 是 $(1-\alpha(2^n))$ -接近的, 而后者又等于 $\Gamma(\langle f_n \rangle)$ 。 Γ 的译码条件断言, 可以利用规模为 $p(n/\alpha(2^n))$ 、有预言帮助的电路 D 来恢复 $\langle f_n \rangle$ 的每一个比特(即计算 f_n), 电路使用(函数) C' 作为预言。结合电路 C' 与 D , 可以得到一个计算 f_n 的、规模为 $p(n/\alpha(2^n)) \cdot S'(n') < S(n)$ 的(标准)电路。定理得证(即与关于 g 的结论矛盾蕴含了与关于 f 的假设矛盾)。

说明

为简化讨论, 我们给出了定义 7.17 的简化陈述, 然而足以证明命题 7.16, 其中 $q(N) = ((\log_2 N)/\alpha(N))^3$ 。问题在于是否存在满足定义 7.17 的编码: 一般来说, 只有使用更详细的译码条件, 并涉及与预言 $\omega: [l(N)] \rightarrow [q(N)]$ 为 $(1 - ((1/q(N)) + \alpha(N)))$ -接近而非

① 该构造还可能得到一个有预言帮助的电路, 能与 ω 几乎 $(1-\alpha(2^n))$ 接近的码字进行译码。也就是说, 存在一个时间复杂度为 $p(n/\alpha(2^n))$ 的概率算法, 能输出一个电路列表, 使得表中以极高概率包含一个有预言帮助的电路, 能与 ω 为 $(1-\alpha(2^n))$ -接近的码字进行译码。进一步地, 以极高概率, 该表中只包含对与 ω 为 $(1-\alpha(2^n)/2)$ -接近的码字进行译码的电路。

$(1-\alpha(N))$ 接近的码字时, 这种编码才有可能存在^①。当然, 当 $\alpha(N) \gg 1/q(N)$ (如命题 7.16) 时, 两者的差别可以忽略, 但对于二数码字 (即 $q(N)=2$, 或在较小字母表上的码字), 差别则比较明显。我们指出, 定理 7.18 可以应用于这种背景下 ($q(N)=2$), 并直接得到一个强不可近似的谓词。细节详见习题 7.21。

7.2.2 对指数规模电路的放大

为了对 BPP 类进行强去随机化, 本小节从一个更强的、关于 ε 的最坏情况电路复杂度假设出发, 并将其转化为更强的不可近似性结论。

定理 7.19 假设 ε 中存在电路复杂度几乎处处为指数的布尔函数 f : 也就是说, 存在常数 $b > 0$, 使得对除有限个 n 之外, 任意能在 $\{0,1\}^n$ 上正确计算 f 的电路其规模至少为 $2^{b \cdot n}$ 。则对某个常数 $c > 0$ 及 $T(n) \stackrel{\text{def}}{=} 2^{c \cdot n}$, ε 中存在 T -不可近似的布尔函数。

在上述假设下 (见定理 8.19 的第 1 部分), 定理 7.19 可用于对 BPP 类进行完全的去随机化 (即 $BPP=P$)。

定理 7.19 实际上是命题 7.16 的一个特例 (与定理 7.8 相结合, 见习题 7.22)。以下简要描述另一种证明, 该证明的思想本身也有一定意义。这种证明从定理 7.12 中提供的温和不可近似谓词出发。然而, 此时无法应用 Yao 的异或引理 (即定理 7.13), 因为后者将近似计算定义于 $\text{poly}(n)$ -比特输入上的谓词的两个电路的规模相关联, 其中一个以较高概率出错, 另一个出错概率不太大。也就是说, Yao 的异或引理声称, 如果 $f: \{0,1\}^{\text{poly}(n)} \rightarrow \{0,1\}$ 可以温和地不可由 S_f -规模的电路求近似, 则 $F: \{0,1\}^{\text{poly}(n)} \rightarrow \{0,1\}$ 由 S_F -规模的电路强不可近似, 其中 $S_F(\text{poly}(n))$ 与 $S_f(n)$ 多项式相关。特别地, $S_F(\text{poly}(n)) < S_f(n)$ 似乎继承了这种推理。对于下界为多项式的情形, 这就足够了 (即如果 S_f 可以是任意大的多项式, 则 S_F 也是), 但当 $S_f(n) = \exp(\Omega(n))$ 时, 不能得到 $S_F(m) = \exp(\Omega(m))$ (而只能得到 $S_F(m) = \exp(m^{\Omega(1)})$)。

困难的根源在于, 对不可近似性的放大是通过多项式数量的独立实例实现的。事实上, 我们无法期望在实现困难放大时不将 f 应用于多个实例, 但是这些实例没有必要相互独立。因此, 主要证明思想是定义 $F(r) = \bigoplus_{i=1}^{\text{poly}(n)} f(x_i)$, 其中 $x_1, x_2, \dots, x_{\text{poly}(n)} \in \{0,1\}^n$ 由 r 生成, 且仍有 $|r| = O(n)$ 。也就是说, 要寻找 Yao 的异或引理的“去随机化”版本。换言之, 要寻找一个“伪随机数发生器”可将 r 扩展到相关的 x_i , 使得对各个 $f(x_i)$ 的异或与独立的 x_i 一样, 是不可近似的。^②

教学注释: 注意到本节其余部分与第 8 章有较强联系。在概念方面, 本节将使用第 8 章的技术工具来证明异或引理的去随机化版本。这些工具包括两两独立的发生器 (见 8.5.1 节)、

① 注意到这是“正确的”定义, 因为当 $\alpha(N) < 1/q(N)$ 时, 似乎不可能满足译码条件 (如定义 7.17 所述)。具体而言, 我们期望 $[q(N)]$ 上一个随机的 $l(N)$ -序列与任意给定码字是 $(1 - (1/q(N)))$ -接近的, 且当 $l(N) \gg q(N^2)$ 时, 以显著优势与几乎所有码字是 $(1 - ((1-o(1))/q(N)))$ -接近的。但当 $N > \text{poly}(q(N))$ 时, 无法希望基于 $\text{poly}(q(N) \log N)$ 比特建议来恢复所有的 N -比特串。

② 这实际上属于 8.1 节的通用模式, 该方法提供了另一种角度来研究随机性与计算困难性之间的联系, 这是第 8 章的重点内容 (见 8.2.7.2 节)。

扩张图上的随机漫游 (见 8.5.3 节) 以及 Nisan-Wigderson 构造 (构造 8.17)。事实上, 前面也曾提到 7.2.2 节属于前沿性内容, 最好留作独立的阅读。

证明的重点是布尔函数的困难区域。不严格地讲, 如果布尔函数 f 对于 S 中随机输入是强不可近似的, 则称 S 是 f 的困难区域, 也就是说, 对任意 (相对) 小的电路 C_n , 有 $\Pr[C_n(U_n) = f(U_n) | U_n \in S] \approx 1/2$ 。由定义可知, $\{0,1\}^*$ 是任意强不可近似谓词的困难区域。正如以下将指出的, 任意的温和不可近似谓词有一个困难区域, 其密度与不可近似因子有关。不严格地讲, 在实现困难放大时, 可以生成以足够高概率属于困难区域的相关实例。但首先需要给出困难区域的概念。

7.2.2.1 困难区域

实际上, 可将困难区域的概念推广到任意分布上。令 X_n 等于 U_n (即 $\{0,1\}^n$ 上的均匀分布), 由定义 7.20 可得到 (n -比特串的) 均匀分布的一个重要特例。一般来说, 只假设 $X_n \in \{0,1\}^n$ 。

定义 7.20 (任意分布的困难区域) 令 $f: \{0,1\}^* \rightarrow \{0,1\}$ 为布尔谓词, $\{X_n\}_{n \in \mathbb{N}}$ 为概率空间, $s: \mathbb{N} \rightarrow \mathbb{N}$ 且 $\varepsilon: \mathbb{N} \rightarrow [0,1]$ 。

• 称集合 S 是 f 的关于 $s(\cdot)$ -规模电路及优势 $\varepsilon(\cdot)$ 的, 相对于 $\{X_n\}_{n \in \mathbb{N}}$ 的困难区域, 如果对任意 n 及任意规模不超过 $s(n)$ 的电路 C_n , 有

$$\Pr[C_n(X_n) = f(X_n) | X_n \in S] \leq \frac{1}{2} + \varepsilon(n)$$

• 称 f 具有相对于 $\{X_n\}_{n \in \mathbb{N}}$ 的密度为 $\rho(\cdot)$ 的困难区域 (关于 $s(\cdot)$ -规模电路及优势 $\varepsilon(\cdot)$), 如果存在集合 S , 是 f 相对于 $\{X_n\}_{n \in \mathbb{N}}$ 的困难区域 (关于上述参数), 使得 $\Pr[X_n \in S_n] \geq \rho(n)$ 。

注意: 布尔函数 f 是 $(s, 1-2\varepsilon)$ -不可近似的当且仅当 $\{0,1\}^*$ 是 f 关于 $s(\cdot)$ -规模电路及优势 $\varepsilon(\cdot)$ 的, 相对于 $\{X_n\}_{n \in \mathbb{N}}$ 的困难区域。因此, 强不可近似性谓词 (如超多项式 S 的 S -不可近似谓词) 具有密度为 1 的困难区域 (关于一个可忽略的优势)。^①但是这一平凡现象无法提供温和和不可近似谓词的困难区域 (关于一个较小的优势 (接近于 0))。以下定理给出了这种困难区域。

定理 7.21 (温和不可近似谓词的困难区域) 令 $f: \{0,1\}^* \rightarrow \{0,1\}$ 为布尔谓词, $\{X_n\}_{n \in \mathbb{N}}$ 为概率空间, $s: \mathbb{N} \rightarrow \mathbb{N}$, $\rho: \mathbb{N} \rightarrow [0,1]$ 满足 $\rho(n) > 1/\text{poly}(n)$ 。假设对任意规模至多为 $s(n)$ 的电路 C_n , 有 $\Pr[C_n(X_n) = f(X_n)] \leq 1 - \rho(n)$ 。对任意 $\varepsilon: \mathbb{N} \rightarrow [0,1]$, 函数 f 具有关于 $s'(\cdot)$ -规模电路及优势 $\varepsilon(\cdot)$ 的, 相对于 $\{X_n\}_{n \in \mathbb{N}}$ 密度为 $\rho'(\cdot)$ 的困难区域, 其中 $\rho'(n) \stackrel{\text{def}}{=} (1 - o(1)) \cdot \rho(n)$, 且 $s'(n) \stackrel{\text{def}}{=} s(n) / \text{poly}(n/\varepsilon(n))$ 。

特别地, 如果 f 是 $(s, 2\rho)$ -不可近似的, 则 f 具有密度为 $\rho'(\cdot) \approx \rho(\cdot)$ 的、相对于均匀分布的困难区域 (关于 $s'(\cdot)$ -规模电路及优势 $\varepsilon(\cdot)$)。

证明概要^②: 首先证明 $\{X_n\}$ 与概率空间 $\{Y_n\}$ “有关” (或者 “支配” 着后者), 而 f 在 $\{Y_n\}$

① 类似地, 温和的不可近似谓词具有密度为 1 的困难区域, 其优势明显小于 1/2。

② 细节见参考文献[102]中附录 A。

上强不可近似, 然后证明这蕴含着断言中的困难区域。事实上, “相关概率空间”的概念在证明中起重要作用。

对于 $\rho: \mathbf{N} \rightarrow [0, 1]$, 如果对任意 x , 有 $\Pr[X_n = x] \geq \rho(n) \cdot \Pr[Y_n = x]$, 则称 $\{X_n\}$ 以 ρ 支配着 $\{Y_n\}$, 也可称 $\{Y_n\}$ 被 $\{X_n\}$ ρ -支配。如果对任意 x , 或者有 $\Pr[Y_n = x] = (1/\rho(n)) \cdot \Pr[X_n = x]$, 或者 $\Pr[Y_n = x] = 0$, 则称 $\{Y_n\}$ 被 $\{X_n\}$ ρ -严格支配。^①

支配及严格支配的概念在证明中起着重要作用。定理的证明包括两部分: 第一部分(断言 7.21.1) 证明对于定理假设中的 $\{X_n\}$ 和 ρ , 存在一个被 $\{X_n\}$ ρ -支配的概率空间 $\{Y_n\}$, 使得 f 在 $\{Y_n\}$ 上是强不可近似的; 第二部分(断言 7.21.2) 证明这个被支配空间的存在蕴含了存在一个被 $\{X_n\}$ ρ' -严格支配的概率空间 $\{Z_n\}$, 使得 f 在 $\{Z_n\}$ 上是强不可近似的。最后, 注意到这样一个被严格支配的空间能得到 f 关于 $\{X_n\}$ 的困难区域, 定理得证。

断言 7.21.1 在定理假设条件下, 存在一个被 $\{X_n\}$ ρ -支配的概率空间 $\{Y_n\}$, 使得对任意 $s'(n)$ 规模的电路 C_n , 有

$$\Pr[C_n(Y_n) = f(Y_n)] \leq \frac{1}{2} + \frac{\varepsilon(n)}{2} \quad (7.13)$$

证明: 用反证法, 假设对任意被 X_n ρ -支配的分布 Y_n , 存在一个规模为 $s'(n)$ 的电路 C_n , 使得 $\Pr[C_n(Y_n) = f(Y_n)] > 0.5 + \varepsilon'(n)$, 其中 $\varepsilon'(n) = \varepsilon(n)/2$ 。一个重要的现象是, 在所有被 X_n ρ -支配的分布集合与所有(被 X_n) 严格 ρ -支配的分布的凸组合集之间存在对应关系, 也就是说, 每个被 ρ -支配的分布是严格 ρ -支配分布的凸组合, 反之亦然(见附录 D.4.1.1 中的特例)。因此, 考虑对(被 X_n) 严格 ρ -支配分布的枚举 $Y_n^{(1)}, Y_n^{(2)}, \dots, Y_n^{(l)}$, 可以得到结论, 对任意 $[l]$ 上的分布 π , 存在一个 $s'(n)$ -规模的电路 C_n , 使得

$$\sum_{i=1}^l \pi(i) \cdot \Pr[C_n(Y_n^{(i)}) = f(Y_n^{(i)})] > 0.5 + \varepsilon'(n) \quad (7.14)$$

现在考虑一个两方参与的游戏, 其中第一方选择一个(被 X_n) 严格 ρ -支配的分布, 第二方选择一个 $s'(n)$ -规模的电路, 并得到由相应成功概率确定的结果, 也就是说, 如果第一方选择第 i 个严格支配的分布, 第二方选择电路 C , 则得到的结果等于 $\Pr[C(Y_n^{(i)}) = f(Y_n^{(i)})]$ 。

式(7.14)可以解释为对于第一方的任意一种随机策略, 存在第二方的一种确定策略, 可得到大于 $0.5 + \varepsilon'(n)$ 的平均结果。Min-Max 原则(见 von Neumann^[235])断言在这种情况下, 无论第一方采取何种策略, 均存在第二方的一种随机策略, 可以得到大于 $0.5 + \varepsilon'(n)$ 的平均结果。这就意味着存在一种 $s'(n)$ -规模电路的分布, 记作 D_n , 满足对任意 i , 有

$$\Pr[D_n(Y_n^{(i)}) = f(Y_n^{(i)})] > 0.5 + \varepsilon'(n) \quad (7.15)$$

其中概率针对着对所有可能的电路 D_n 以及随机变量 Y_n 。令 $B_n = \{x: \Pr[D_n(x) = f(x)] \leq 0.5 + \varepsilon'(n)\}$, 则 $\Pr[X_n \in B_n] < \rho(n)$, 因为否则可以定义 Y_n , 使得当 $x \in B_n$ 时 $\Pr[Y_n = x] = \Pr$

^① 实际上, 应该允许一种例外, 即对上述条件适当放宽, 要求对至多一个串 $x \in \{0, 1\}^n$, 有 $0 < \Pr[Y_n = x] < \Pr[X_n = x]/\rho(n)$ 。这对于证明几乎不产生什么影响, 因此我们忽略了这种例外情形。

$[X_n = x] / \Pr[X_n \in B_n]$, 否则, $\Pr[Y_n = x] = 0$, 从而与式 (7.15) 矛盾^①。利用对 D_n 的标准放大, 可得到 $\text{poly}(n/\varepsilon'(n)) \cdot s'(n)$ -规模电路的一种分布 D'_n , 使得对任意 $x \in \{0,1\}^n \setminus B_n$, 有 $\Pr[D'_n(x) = f(x)] > 1 - 2^{-n}$, 从而存在一个 $s(n)$ -规模的电路 C_n , 使得对任意 $x \in \{0,1\}^n \setminus B_n$, 有 $C_n(x) = f(x)$, 这蕴含了 $\Pr[C_n(X_n) = f(X_n)] \geq \Pr[X_n \in \{0,1\}^n \setminus B_n] > 1 - \rho(n)$, 与假设矛盾。定理得证。□

下面证明, 当只允许 (被 $\{X_n\}$) ρ -严格支配的概率空间时, 断言 7.21.1 的结论 (被叙述为由 $\{X_n\}$ ρ -支配的概率空间) 也成立。以下的正式描述涉及到支配参数 ρ (以及优势 ε) 在某种程度上的丢失。

断言 7.21.2 如果存在一个被 $\{X_n\}$ ρ -支配的概率空间 $\{Y_n\}$, 使得对任意 $s'(n)$ -规模的电路 C_n , 有 $\Pr[C_n(Y_n) = f(Y_n)] \leq 0.5 + (\varepsilon(n)/2)$, 则存在一个被 $\{X_n\}$ 严格 ρ' -支配的概率空间 $\{Z_n\}$, 使得对任意 $s'(n)$ -规模的电路 C_n , 有 $\Pr[C_n(Z_n) = f(Z_n)] \leq 0.5 + \varepsilon(n)$ 。

换言之, 断言 7.21.2 称函数 f 具有相对于 $\{X_n\}$ 的、关于 $s'(\cdot)$ -规模电路和优势 $\varepsilon(\cdot)$ 的、密度为 $\rho'(\cdot)$ 的困难区域, 从而证明了定理。断言 7.21.2 的证明使用概率论方法 (见参考文献 [11])。具体而言, 随机选择一个集合 S_n , 其中包含每个 n -比特串 x 的概率为

$$p(x) \stackrel{\text{def}}{=} \frac{\rho(n) \cdot \Pr[Y_n = x]}{\Pr[X_n = x]} \leq 1 \quad (7.16)$$

独立于其他串的选择, 可以证明, 对所有可能的 S_n , $\Pr[X_n \in S_n] \approx \rho(n)$ 以极高概率成立, 且对任意规模为 $s'(n)$ 的电路 C_n , 有 $\Pr[C_n(X_n) = f(X_n) | X_n \in S_n] < 0.5 + \varepsilon(n)$ 。证明后一个断言时, 可对所有相关的电路求联合界限, 证明对所有这样的电路 C_n , 以及所有可能的 S_n , $|\Pr[C_n(X_n) = f(X_n) | X_n \in S_n] - \Pr[C_n(Y_n) = f(Y_n)]| < \varepsilon(n)/2$ 成立的概率为 $1 - \exp(-s'(n)^2)$ 。

细节见参考文献 [102] 中附录 A。这就完成了定理的证明。□

7.2.2.2 通过困难区域实现困难放大

在阐述如何利用困难区域的概念来证明 Yao 异或引理的去随机化版本时, 首先要说明如何利用它证明该引理的原始版本 (即定理 7.13)。

Yao 异或引理的另一种证明

令 f 、 p_1 和 p_2 如定理 7.13, 则由定理 7.21 可知, 对于 $\rho'(n) = 1/3p_2(n)$ 以及 $s'(n) = p_1(n)^{\Omega(1)} / \text{poly}(n)$, 函数 f 有关于 $s'(\cdot)$ -规模电路和优势 $1/s'(\cdot)$ 的、密度为 ρ' 的困难区域 S (相对于 $\{U_n\}$)。因此, 对于定理 7.13 中的 $t(n) = n \cdot p_2(n)$ 及 F , 在 F 的 $t(n)$ 个随机 n -比特块中, 有一个属于 S (即 F 的困难区域) 的概率至少为 $1 - (1 - \rho'(n))^{t(n)} = 1 - \exp(-\Omega(n))$ 。直观上, 这足以证明 F 的强不可近似性。事实上, 用反证法, 假设有一个小规模电路 C_n (规模为 $p'(t(n) \cdot n)$), 以优势 $\varepsilon(n) + \exp(-\Omega(n))$ 对 F (在 $U_{t(n) \cdot n}$ 上) 求近似, 其中 $\varepsilon(n) > 1/s'(n)$, 则 $\varepsilon(n)$ 项的存在必定是由于 $t(n) \cdot n$ -比特的输入中包含了 S 中的一个块。利用平均化方法,

① 注意: Y_n 是被 X_n ρ -支配的, 而由假设可知, $\Pr[D_n(Y_n) = f(Y_n)] \leq 0.5 + \varepsilon'(n)$ 。利用任意 ρ -支配的分布是严格 ρ -支配分布的凸组合这一事实, 对某个严格 ρ -支配的 $Y_n^{(i)}$, 有 $\Pr[D_n(Y_n^{(i)}) = f(Y_n^{(i)})] \leq 0.5 + \varepsilon'(n)$ 。

首先固定该块的索引, 再固定其他块的索引, 并得到如下推断: 对某个 $i \in [t(n)]$ 及 $x_1, x_2, \dots, x_{i(n)} \in \{0, 1\}^n$, 有

$$\Pr[C_n(x', U_n, x'') = F(x', U_n, x'') | U_n \in S] \geq \frac{1}{2} + \varepsilon(n)$$

其中 $x' = (x_1, x_2, \dots, x_{i-1}) \in \{0, 1\}^{(i-1)n}$, $x'' = (x_{i+1}, x_{i+2}, \dots, x_{i(n)}) \in \{0, 1\}^{(i(n)-i)n}$ 。如果在 C_n 中硬性嵌入 $i \in [t(n)]$, $x' = (x_1, x_2, \dots, x_{i-1})$, $x'' = (x_{i+1}, x_{i+2}, \dots, x_{i(n)})$ 以及 $\sigma = \bigoplus_{j \neq i}^{\text{def}} f(x_j)$, 则得到与 S 是 f 的困难区域这一事实 (已证明) 相矛盾的结论 (利用电路 $C'_n(z) = C_n(x', z, x'') \oplus \sigma$), 从而定理得证 (对 $p'(t(n) \cdot n) \leq s'(n) - 1$)。

Yao 异或引理的去随机化版本

首先阐明如何利用困难区域将非常温和的不可近似性放大为常数级的不可近似性。回顾我们的目标是将给定函数应用于多个 (相关) 实例, 以得到这样一种放大, 其中每个实例的长度与目标函数输入长度呈线性关系。事实上, 这些相关实例可以用一个充分的“伪随机数发生器” (见第 8 章) 产生。以下的放大过程利用两两独立的发生器 (见 8.5.1 节), 记作 G , 其中将 $2n$ -比特的种子扩展为 n 个串, 每串长度为 n 。

引理 7.22 (到达常数级不可近似性的去随机异或引理) 设 $f: \{0, 1\}^* \rightarrow \{0, 1\}$ 是 (T, ρ) -不可近似的, 其中 $\rho(n) > 1/\text{poly}(n)$, 并为了简单, 假设 $\rho(n) \leq 1/n$ 。令 b 表示内积模 2 谓词, G 为上述的两两独立发生器, 则 $F_1(s, r) = b(f(x_1)f(x_2)\cdots f(x_n), r)$, 其中 $|r| = n = |s|/2$ 且 $(x_1, x_2, \dots, x_n) = G(s)$ 是 (T', ρ') -不可近似的, 其中 $T'(n') = T(n'/3)/\text{poly}(n')$ 及 $\rho'(n') = \Omega(n' \cdot \rho(n'/3))$ 。

当然, 如果 $f \in \mathcal{E}$, 则 $F_1 \in \mathcal{E}$ 。应用引理 7.22 常数次, 可将一个 $(T, 1/\text{poly})$ -不可近似谓词转化为一个 $(T'', \Omega(1))$ -不可近似谓词, 其中 $T''(n'') = T(n''/O(1))/\text{poly}(n'')$ 。

证明概要: 与前面的证明相同 (Yao 异或引理的原始版本), 首先应用定理 7.21。在设置参数时, 使得对于 $\alpha(n) = \rho(n)/3$ 及 $t'(n) = T(n)/\text{poly}(n)$, 函数 f 具有密度为 α 的, 关于 $t'(\cdot)$ 规模电路及优势 0.01 的困难区域 S (相对于 $\{U_n\}$)。下面, 如 7.2.1.2 节所示, 考虑相应的 (去随机化) 直积问题, 即函数 $P_1(S) = (f(x_1), f(x_2), \dots, f(x_n))$, 其中 $|s| = 2n$ 且 $(x_1, x_2, \dots, x_n) = G(s)$ 。首先证明 P_1 对其定义域中 $\Omega(n \cdot \alpha(n))$ 比例的输入是难计算的, 从而证明了 F_1 的量化不可近似性。

一个重要的现象是, 由习题 7.23 可知, $G(U_{2n})$ 输出的 n 个串中至少有一个属于 S 的概率不小于 $\beta(n) = n \cdot \alpha(n)/2$ 。直观上, 我们期望每个 $t'(n)$ -规模的电路无法计算 $P_1(U_{2n})$ 的概率至少为 $0.49\beta(n)$, 这是因为序列 $G(U_{2n})$ 包含 f 的困难区域中一个元素的概率是 $\beta(n)$ (此时, 这个值被正确猜测的概率至多为 0.51)。实际证明依赖于可约性方法, 它不像非去随机化情形中那么直接。

出于技术上的原因^①, 我们利用条件 $\alpha(n) < 1/2n$ (这可由 $\rho(n) \leq 1/n$ 的假设来保证,

① 以下讨论将基于 $\beta(n) - \gamma(n) > 0.51n \cdot \alpha(n)$ 的事实, 其中 $\gamma(n) = \Omega(\beta(n))$ 。

而这里令 $\alpha(n) = \rho(n)/3$ 。此时, 习题 7.23 蕴含了 $G(U_{2n})$ 输出的 n 个串中至少有一个属于 S 的概率不小于 $\beta(n) \stackrel{\text{def}}{=} 0.75 \cdot n \cdot \alpha(n)$ 。以下证明每个 $(t'(n) - \text{poly}(n))$ -规模电路计算 P_1 的失败概率至少是 $\gamma(n) = 0.3\beta(n)$ 。照例可利用归纳法。令 $G(s)_i$ 表示序列 $G(s)$ 的第 i 个串(即 $G(s) = (G(s)_1, G(s)_2, \dots, G(s)_n)$), 并注意到, 给定 i 及 x , 可以高效地采样 $G_i^{-1}(x) \stackrel{\text{def}}{=} \{s \in \{0,1\}^{2n} : G(s)_i = x\}$ 。给定一个以 $1 - \gamma(n)$ 的成功概率计算 $P_1(U_{2n})$ 的电路 C_n , 考虑电路 C'_n , 对于输入 x , 均匀选择 $i \in [n]$ 及 $s \in G_i^{-1}(x)$, 并输出 $C_n(s)$ 的第 i 个比特, 则由 (C'_n) 的构造及关于 C_n 的假设, 有

$$\begin{aligned} \Pr[C'_n(U_n) = f(U_n) | U_n \in S] &\geq \sum_{i=1}^n \frac{1}{n} \cdot \Pr[C_n(U_{2n}) = P_1(U_{2n}) | G(U_{2n})_i \in S] \\ &\geq \frac{\Pr[C_n(U_{2n}) = P_1(U_{2n}) \wedge \exists i G_i(U_{2n})_i \in S]}{n \cdot \max_i \{\Pr[G(U_{2n})_i \in S]\}} \\ &\geq \frac{(1 - \gamma(n)) - (1 - \beta(n))}{n \cdot \alpha(n)} \\ &= \frac{0.7\beta(n)}{n \cdot \alpha(n)} > 0.52 \end{aligned}$$

这与 S 是 f 的关于 $t'(\cdot)$ -规模电路及优势 0.01 的困难区域这一事实矛盾, 从而证明了任意 $(t'(n) - \text{poly}(n))$ -规模电路在计算 P_1 时, 失败概率至少是 $\gamma(n) = 0.3\beta(n)$ 。

证明了 P_1 的困难性之后, 现在要推导 F_1 的温和不可近似性, 其中 $F_1(s, r) = b(P_1(s), r)$ 。

这里利用定理 7.7 证明中一开始讨论的简单情况(其中预言者的错误概率小于 $1/4$, 而不是像完全结论中那样, 要求预言错误概率只小于 $1/2$)就足够了。将电路 C 预言错误的概率记作 $\eta_C(s) = \Pr_{r \in \{0,1\}^n} [C(s, r) \neq b(P_1(s), r)]$, 回顾如果 $\eta_C(s) \leq 0.24$, 则可以利用 C 来恢复 $P_1(s)$ 。因此, 对于规模为 $T'(3n) = t'(n)/\text{poly}(n)$ 的电路 C , 一定有 $\Pr_s [\eta_C(s) > 0.24] \geq \gamma(n)$, 从而有 $E_s [\eta_C(s)] > 0.24\gamma(n)$, 这意味着任意 $T'(3n)$ -规模的电路计算 $(s, r) \mapsto b(P_1(s), r)$ 的失败概率至少是 $\delta(|s| + |r|) \stackrel{\text{def}}{=} 0.24 \cdot \gamma(|r|)$ 。这意味着 F_1 是 $(T', 2\delta)$ -不可近似的, 从而引理得证(注意到 $\delta(n') = \Omega(n' \cdot \alpha(n'/3))$)。□

以下引理将常数不可近似性放大为强不可近似性。事实上, 将定理 7.12 与引理 7.22 及 7.23 相结合, 可以得到定理 7.19 (作为一个特例)。

引理 7.23 (始于常数不可近似性的去随机异或引理) 假设 $f: \{0,1\}^* \rightarrow \{0,1\}$ 是 (T, ρ) -不可近似的, 其中 ρ 为常数, b 表示内积模 2。则存在一个指数时间可计算的函数 G , 使得 $F_2(s, r) = b(f(x_1)f(x_2)\cdots f(x_n), r)$ 是 T' -不可近似的, 其中 $(x_1, x_2, \dots, x_n) = G(s)$ 且 $n = \Omega(|s|) = |r| = |x_1| = \cdots = |x_n|$, 其中 $T'(n') = T(n'/O(1))^{\Omega(1)}/\text{poly}(n')$ 。

如果 $f \in \mathcal{E}$, 则仍有 $F \in \mathcal{E}$ 。

证明概要：^①与引理 7.22 相同，先构造 f 的一个密度为 $\rho/3$ 的困难区域（关于 $T(n)^{\Omega(1)}/\text{poly}(n)$ - 规模电路及优势 $T(n)^{-\Omega(1)}$ ），并主要分析对应于计算函数 $P_2(s) = (f(x_1), f(x_2), \dots, f(x_n))$ 的（去随机化）直积问题，其中 $|s| = O(n)$ 且 $(x_1, x_2, \dots, x_n) = G(s)$ 。“发生器” G 被定义为满足 $G(s's'') = G_1(s') \oplus G_2(s'')$ ，其中 $|s'| = |s''|$ ， $|G_1(s')| = |G_2(s'')|$ ，且满足以下条件。

(1) G_1 是 8.5.3 节讨论的扩张随机漫游生成器。可以证明， $G_1(U_{O(n)})$ 输出由 n 个串构成的序列，使得对任意密度为 ρ 的集合 S ，至少有 $\Omega(\rho n)$ 个串属于 S 的概率为 $1 - \exp(-\Omega(\rho n))$ 。注意：如果对任意 $|s'| = |s''|$ ，有 $|G_1(s')| = |G_2(s'')|$ ，则 G 也具备上述性质，从而，在 P_2 定义域中存在常数比例的 x_i 属于 f 困难区域的概率是 $1 - \exp(-\Omega(\rho n))$ 。

小规模电路无法计算 P_2 的概率大于 $\exp(-\Omega(\rho n))$ ，这种说法很吸引人，但是显然只有当属于困难区域的各个 x_i 独立（且均匀）分布时这一点才成立，而这个条件很难实现。所以，实际上，我们利用 G_2 来处理这一问题。

(2) G_2 是构造 8.17 中的“集合投影”系统；具体地， $G_2(s) = (s_{S_1}, s_{S_2}, \dots, s_{S_n})$ ，其中每个 S_i 是 $[s]$ 的一个 n 元素子集，且这些 S_i 两两相交，交集的势至多为 $n/O(1)$ ^②。可以利用类似于定理 8.18 证明中的分析来证明，当给定了 x_i 及所有其他的 $f(x_j)$ 时， x_i 之间的相关性对于计算某个特定 $f(x_i)$ 毫无帮助（注意： G 继承了 G_2 的相关性）。

在对构造的实际分析（通过参考文献[128]中第 3 节的猜测游戏）中，将计算 P_2 的成功概率与在困难区域中猜测 f 的优势相关联。感兴趣的读者可见参考文献[128]。□

思考

证明引理 7.22 和引理 7.23 时，都要先证明相应“直积”引理的去随机化版本（定理 7.14）。事实上，这些证明的核心是证明充分去随机化的“直积”引理。请读者注意这一步的关键作用（特别是在证明引理 7.23 时）：不能将 $f(x_1), f(x_2), \dots, f(x_n)$ 看作是相互独立的（至少对于引理中要求的发生器 G 而言），所以，要尽量避免分析能正确求这些值异或的概率。相反，已经证明正确计算所有的 n 个值是很困难的，所以对这些值的一个随机子集求异或可以得到一个强不可近似谓词（注意：习题 7.17 的方法在这里无法使用，因为各个 x_i 并不独立，这也正是我们对这些值的一个随机子集求异或，而不是对全部值求异或的原因）。

本章注释

单向函数的概念由 Diffie 和 Hellman 提出^[66]。弱单向函数及单向函数的放大（即定理 7.5）则由 Yao 提出^[239]。定理 7.5 的证明首次出现在参考文献[87]中。

困难核心谓词的概念最早由 Blum 和 Micali 提出^[41]。他们还证明，假设“DLP 函数”（即有限域中的指数）是单向的，则其困难核心是一种特殊的谓词。一般的困难核心谓词（定理

① 细节见参考文献[128]。

② s_S 表示 s 在系数 $S \subseteq [s]$ 上的投影，即对 $s = \sigma_1 \sigma_2 \dots \sigma_k$ 和 $S = \{i_j : j = 1, 2, \dots, n\}$ ，有 $s_S = \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_n}$ 。

7.7) 由 Levin 提出, 并被 Goldreich 和 Levin 证明^[99]。本章描述的证明则由 Rackoff 提出。我们指出, 原始证明也有自身的优点 (见参考文献[105])。

正则去随机发生器的构造 (见 8.3 节), 特别是 Nisan-Wigderson 框架 (构造 8.17) 是研究 ε 中谓词不可近似性的本质原因。定理 7.10 来自参考文献[22], 而定理 7.19 来自参考文献[128]。两个结论都在很大程度上依赖于 Yao 的异或引理, 下面回顾这个引理。

与 Yao 论文^[239]中其他几个基本观点^①相似, Yao 的异或引理 (定理 7.13) 甚至没有出现在参考文献[239]中, 而是来自于 Yao 对其论文的口头阐述。Yao 异或引理第一个正式公开证明应归功于 Levin (见参考文献[102]中第 3 节)。7.2.1.2 节中的证明由 Goldreich、Nisan 和 Wigderson 给出 (见参考文献[102]中第 5 节)。关于 Yao 的异或引理 (及其变形), 最近也出现了其他一些证明方法, 参考文献[223]中给出了这些证明的简要综述, 感兴趣的读者可以参考。

Impagliazzo^[126]最早提出了困难区域的概念 (还可见于参考文献[102]中第 4 节), 并将其用于证明 Yao 异或引理的原始版本。该引理的第一种去随机化版本 (引理 7.22) 也来自参考文献[126], 第二种去随机化 (引理 7.23) 及定理 7.19 来自 Impagliazzo 和 Wigderson^[128]。

最坏情况到平均情况的归约 (即 7.2.1.1 节定理 7.12) 来自参考文献[22]。归约中遵循了 Blum、Luby 和 Rubinfeld^[40]的自校正模式, Lipton^[157]首次利用这种模式构造了 (严格的)^②最坏情况到平均情况的归约^③。

Sudan、Trevisan 和 Vadhan^[218]利用查表译码与困难放大之间的关系 (7.2.1.3 节) 构造了定理 7.10 和定理 7.19 的另一种证明^[218]。

$\mathcal{NP}$ 类的困难放大最近引起人们关注: 参考文献[121]中提出了由温和和不可近似性到强不可近似性的放大, 参考文献[43]中则指出, 最坏情况到平均情况的归约 (至少在非适应性归约下) 是不可行的。

习 题

7.1 证明如果单向函数存在, 则存在保持长度的单向函数 (即对任意 $x \in \{0,1\}^n$, 有 $|f(x)| = |x|$)。

提示: 显然存在某些多项式 p , 使 $|f(x)| < p(|x|)$ 对所有 x 都成立。不失一般性, 假设 $n \mapsto p(n)$ 是单调递增的双射, 并当 $p(n) \leq m < p(n+1)$ 时, 令 $p^{-1}(m) = n$ 。定义 $f'(z) = f(x)0^{1^{|z|-|f(x)|-1}}$, 其中 x 是 z 的 $p^{-1}(|z|)$ -比特前缀。

7.2 证明如果函数 f 在定义 7.3 的意义下是难求逆的, 则在定义 7.1 的意义下也是难求

① 最著名的就是伪随机性与不可预测性的等价关系 (见 8.2.5 节)。

② 自校正模式早期被用于“双论据问题”, 其中包含一个固定论据和一个随机论据 (见参考文献[108])。例如, 考虑这样一个判定问题, 给定 (N, r) , 判定 $x^2 \equiv r \pmod N$ 是否有整数解, 以及从 (N, r) 到 (N, r') 的随机映射, 其中 $r' = r \cdot \omega^2 \pmod N$, 且 ω 在 $[N]$ 中均匀分布。不严格地, 这个过程将得到最坏情况复杂度到“混合最坏/平均情况”复杂度的归约 (或从“混合最坏/平均情况”到单纯平均情况的归约)。

③ 自校正模式的一种早期应用是参考文献[19]中的严格最坏情况到平均情况的归约, 但是其中只涉及低复杂性类。具体地, 这种归约针对着 $\mathcal{AC}^0$ 中可计算的奇偶校验函数 (暗示了奇偶校验不能用 $\mathcal{AC}^0$ 求近似, 因为它不能在该类中计算 (见参考文献[83, 240, 115]))。(随机) 归约将 $x \in \{0,1\}^n$, 视为序列 $(x_1, x_2, \dots, x_n)$, 映射为序列 $x' = (x_1 \oplus r_1, x_1 \oplus x_2 \oplus r_2, x_2 \oplus x_3 \oplus r_3, \dots, x_{n-1} \oplus x_n \oplus r_n)$,

其中 $r_1, r_2, \dots, r_n \in \{0,1\}$ 是独立均匀分布的。注意: x' 在 $\{0,1\}^n$ 上均匀分布, 且 $\text{parity}(x) = \text{parity}(x') \oplus r_n$ 。

逆的。

提示：考虑使式 (7.1) 中概率取最大值的随机数序列。

7.3 假设单向函数存在，证明存在一个不具备强单向性的弱单向函数。

7.4 (通用的单向函数 (由 L. Levin 提出)) 利用通用机的概念，构造一个多项式时间可计算的函数，当且仅当单向函数存在时，该函数是难求逆的 (在定义 7.1 的意义下)。

提示：考虑函数 F ，其输入被解析为一个对 (M, x) ，并模仿 M 对于输入 x 的 $|x|^3$ 步计算。注意：如果存在一个可在立方时间内被模仿的单向函数，则 F 是一个弱单向函数。利用填充证明当且仅当单向函数存在时，存在可在立方时间内被模仿的单向函数。

7.5 对于 $l > 1$ ，证明以下 $2^l - 1$ 个采样两两独立且在 $\{0, 1\}^n$ 上均匀分布。这些采样通过均匀、独立地在 $\{0, 1\}^n$ 中选择 l 个串而得到。将这些串记作 $s^1, s^2, \dots, s^l$ ，生成对应于 $\{1, 2, \dots, l\}$ 的不同非空子集的 $2^l - 1$ 个采样，使得对于子集 J ，令 $r^J \stackrel{\text{def}}{=} \bigoplus_{j \in J} s^j$ 。

提示：对于 $J \neq J'$ ，有 $r^J \oplus r^{J'} = \bigoplus_{j \in K} s^j$ ，其中 K 表示 J 与 J' 的对称差，见 8.5.1 节中的相关内容。

7.6 (对定理 7.7 证明的变形) 证明对任意 $x \in \{0, 1\}^n$ ，当给定满足式 (7.6) 的 $B_x: \{0, 1\}^n \rightarrow \{0, 1\}$ 的预言访问时，该过程在运行 $\text{poly}(n/\varepsilon)$ 步后输出一个串列表，其中包含 x 的概率至少是 $1/2$ 。

7.7 (定理 7.8 的证明) 回顾定理 7.7 的证明实际上证实了存在一个 $\text{poly}(n/\varepsilon)$ -时间预言机 M ，使得对任意满足 $\Pr_r[B(r) = b(x, r)] \geq \frac{1}{2} + \varepsilon$ 的 $B: \{0, 1\}^n \rightarrow \{0, 1\}$ 及 $x \in \{0, 1\}^n$ ，有 $\Pr[M^B(n, \varepsilon) = x] = \Omega(\varepsilon^2/n)$ 。证明这蕴含了定理 7.8 (事实上，利用习题 7.6 可得到另一种证明)。

提示：应用“优惠券收集者”方法。

7.8 一个多项式时间可计算的谓词 $b: \{0, 1\}^* \rightarrow \{0, 1\}$ 被称为通用的困难核心谓词，如果对任意单向函数 f ，谓词 b 是 f 的困难核心。注意：定理 7.7 中的谓词是“几乎通用的” (即对任意单向函数 f ，该谓词是 $f'(x, r) = (f(x), r)$ 的困难核心，其中 $|x| = |r|$)。证明不存在通用的困难核心谓词。

提示：设 b 为通用的困难核心谓词，并令 f 为任意一个单向函数。考虑函数 $f'(x) = (f(x), b(x))$ 。

7.9 证明如果 $\mathcal{NP}$ 不包含于 P/poly ，则 ε 也不包含于 P/poly 中。进一步地，对任意 $S: \mathbb{N} \rightarrow \mathbb{N}$ ，如果 $\mathcal{NP}$ 中某个问题不存在规模为 S 的求解电路，则对某个常数 $\varepsilon > 0$ ， ε 中存在一个问题，不存在规模为 S' 的求解电路，其中 $S'(n) = S(n^\varepsilon)$ 。证明这个结论是“几乎处处”成立的。

提示：虽然尚不知道 $\mathcal{NP}$ 是否包含于 ε 中，但是 SAT 确实属于 ε ，这暗示了 $\mathcal{NP}$ 可归约为 ε 中某个问题。对于“几乎处处”的情况，考虑归约可能不保持输入长度的事实。

7.10 对任意函数 $f: \{0, 1\}^n \rightarrow \{0, 1\}$ ，构造一个线性规模电路 C_n ，满足 $\Pr[C(U_n) = f(U_n)] \geq 0.5 + 2^{-n}$ 。进一步地，对任意 $t \leq 2^{n-1}$ ，构造一个规模为 $O(t \cdot n)$ 的电路

C_n , 满足 $\Pr[C(U_n) = f(U_n)] \geq 0.5 + t \cdot 2^{-n}$ 。注意: 不能假设 $\Pr[f(U_n) = 1] = 0.5$ 。

7.11 (低次多项式的自校正) 设 d, m 为整数, F 为有限域, 其中元素个数大于 $t = dm + 1$ 。
令 $p: F^m \rightarrow F$ 为 d 次多项式, 且 $\alpha_1, \alpha_2, \dots, \alpha_t$ 为 F 中 t 个非零元素。

(1) 证明对任意 $x, y \in F^m$, 可由 $p(x + \alpha_1 y), \dots, p(x + \alpha_t y)$ 的值高效计算 $p(x)$, 其中 x 和 y 被看作是 F 上的 m -维向量。

(2) 证明对任意 $x \in F^m$ 及 $\alpha \in F \setminus \{0\}$, 如果均匀选择 $r \in F^m$, 则点 $x + \alpha r$ 在 F^m 上均匀分布。

最后得出结论, $p(x)$ 可由 t 个随机点来恢复, 其中每个点在 F^m 上均匀分布。

7.12 (低次扩展) 证明对任意 $H \subset F$ 及任意函数 $f: H^m \rightarrow F$, 存在一个 m -变量 $|H| - 1$ 次多项式 $\hat{f}: H^m \rightarrow F$, 满足对任意 $x \in H^m$, 有 $\hat{f}(x) = f(x)$ 。

提示: 定义 $\hat{f}(x) = \sum_{a \in H^m} \delta_a(x) \cdot f(a)$, 其中 δ_a 为一个 m -变量 $|H| - 1$ 次多项式, 满足 $\delta_a(a) = 1$, 而对任意 $x \in H^m \setminus \{a\}$, $\delta_a(x) = 0$ 。特别地, $\delta_{a_1 a_2 \dots a_m}(x_1, x_2, \dots, x_m) = \prod_{i=1}^m \prod_{b \in H \setminus \{a_i\}} ((x_i - b)/(a_i - b))$ 。

7.13 设 $\hat{f}$ 和 S' 如定理 7.12 结论中所述, 证明 ε 中存在布尔函数 g , 对于 $S''(n' + O(\log n')) = S'(n')/n'$ 和 $\varepsilon(m) = 1/m^3$, 是 (S'', ε) -不可近似的。

提示: 考虑这样定义的函数 g , 使得 $g(x, i)$ 等于 $\hat{f}(x)$ 的第 i 个比特。

7.14 (定理 7.8 的典型应用) 对任意 $l: \mathbb{N} \rightarrow \mathbb{N}$, 设 $h: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 为任意一个函数 (或随机映射), 满足对任意 $x \in \{0, 1\}^*$, 有 $|h(x)| = l(|x|)$, $\{X_n\}_{n \in \mathbb{N}}$ 为一个概率空间。假设对某个 $s: \mathbb{N} \rightarrow \mathbb{N}$ 及 $\varepsilon: \mathbb{N} \rightarrow (0, 1]$, 任意 s -规模的电路类及足够大的 n , 有 $\Pr[C_n(X_n) = h(X_n)] \leq \varepsilon(n)$ 。假设 $s': \mathbb{N} \rightarrow \mathbb{N}$ 和 $\varepsilon': \mathbb{N} \rightarrow (0, 1]$ 满足 $s'(n + l(n)) \leq s(n)/\text{poly}(n/\varepsilon'(n + l(n)))$ 且 $\varepsilon'(n + l(n)) = \Omega(n) \cdot \varepsilon(n)^{\Omega(1)}$ 。证明定理 7.8 蕴含了对任意 s' -规模的电路簇 $\{C'_n\}_{n' \in \mathbb{N}}$ 及所有足够大的 $n' = n + l(n)$, 有

$$\Pr[C'_{n+l(n)}(X_n, U_{l(n)}) = b(h(X_n), U_{l(n)})] \leq \frac{1}{2} + \varepsilon'(n + l(n))$$

其中 $b(y, r)$ 表示 y 与 r 模 2 的内积。注意: 如果 X_n 在 $\{0, 1\}^n$ 上均匀分布, 则谓词 $h'(x, r) = b(h(x), r)$, 其中 $|r| = |h(x)|$ 是 $(s', 1 - 2\varepsilon')$ -不可近似的。此时, 可以得到结论, 如果 $\varepsilon(n) = 1/s(n)$ 且 $s'(n + l(n)) = s(n)^{\Omega(1)}/\text{poly}(n)$, 则 h' 是 s' -不可近似的。

7.15 (习题 7.14 的逆命题(由 Voila 和 Wigderson 给出)) 设 $l: \mathbb{N} \rightarrow \mathbb{N}$, $h: \{0, 1\}^* \rightarrow \{0, 1\}^*$, $\{X_n\}_{n \in \mathbb{N}}$ 和 b 如习题 7.14 所述。令 $H(x, r) = b(h(x), r)$, 并回顾在习题 7.14 中, 将猜测 h 归约为求 H 的近似。现在, 构造一个反向的归约。即证明如果 H 是 $(s, 1 - \varepsilon)$ -不可近似的 (在 $\{X_n\}_{n \in \mathbb{N}}$ 上), 则任意 s' -规模的电路计算 h 的成功概率不超过 ε , 其中 $s'(n) = s(n) - O(l(n))$ 。

提示: 照例先假设存在一个 s' -规模的电路, 以大于 ε 的成功概率计算 h 。考虑两个相关的分布于 $\{0, 1\}^{l(n)}$ 上的随机变量 X 和 Y , 其中 X 表示 $h(U_n)$ 的取值, Y 表示电路对该值的猜测。

证明对于一个均匀分布的 $r \in \{0,1\}^{l(n)}$, 有 $\Pr[b(X,r)=b(Y,r)]=(1+p)/2$, 其中 $p \stackrel{\text{def}}{=} \Pr[X=Y]$ 。

7.16 (利用平均法的去随机化) 令 $C:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^l$ 表示一个“概率电路”, 在处理第一个输入时在使用一串随机序列, 该序列就是第二个输入。令 X 和 Z 分别为两个分布于 $\{0,1\}^n$ 和 $\{0,1\}^m$ 上的独立随机变量, 且令 χ 为布尔谓词 (表示关于 C 行为成功的事件)。证明存在一个串 $z \in \{0,1\}^m$, 使得对 $C_z(x) \stackrel{\text{def}}{=} C(x,z)$, 有 $\Pr[\chi(X, C_z(X))=1] \geq \Pr[(X, C(X, Z))=1]$ 。

7.17 (将“选择性异或”归约为“标准异或”) 令 f 为布尔函数, $b(y,r)$ 表示等长串 y 与 r 的内积模 2。假设 $F'(x_1, x_2, \dots, x_{t(n)}, r) \stackrel{\text{def}}{=} b(f(x_1)f(x_2)\cdots f(x_{t(n)}), r)$, 其中, $x_1, x_2, \dots, x_{t(n)} \in \{0,1\}^n$ 且 $r \in \{0,1\}^n$ 是 T' -不可近似的。设 $n \mapsto t(n) \cdot n$ 为双射, 证明 $F(x) \stackrel{\text{def}}{=} F'(x, 1^{t(|x|)})$ 是 T -不可近似的, 其中 $t'(t(n) \cdot n) = t(n)$, $T(m) = T'(m + t'(m)) - O(m)$ 。

提示: 将对 F' 的近似问题归约为对 F 求近似。关键在于对任意 $x = (x_1, x_2, \dots, x_{t(n)})$, $x' = (x'_1, x'_2, \dots, x'_{t(n)})$ 及 $r = r_1 r_2 \cdots r_{t(n)}$, 满足当 $r_i = 1$ 时 $x'_i = x_i$, 则有 $F'(x, r) = F(x') \oplus \bigoplus_{i:r_i=0} f(x'_i)$ 。这里暗含了一个由 F' 到 F 的非一致性归约, 其中使用“充分的” $z = z_1 z_2 \cdots z_{t(n)} \in \{0,1\}^n$ 以及相应的 $f(z_i)$ 值作为建议。对于输入 $x_1, x_2, \dots, x_{t(n)}$, $r_1 r_2 \cdots r_{t(n)}$, 当 $r_i = 1$ 时, 在归约中令 $x'_i = x_i$, 否则, 令 $x'_i = z_i$, 发出向 F 的询问 $x' = (x'_1, \dots, x'_{t(n)})$, 并返回 $F(x') \oplus \bigoplus_{i:r_i=0} f(z_i)$ 。当 $z_1, z_2, \dots, z_{t(n)} \in \{0,1\}^n$ 为均匀分布时, 分析这个归约, 并得出结论, 可将其设置为某个固定值 (见习题 7.16)。^①

7.18 (将“标准异或”归约为“选择性”异或) 继续习题 7.17, 构造一个反向的归约。即对于上述的 F 及 F' , 证明如果 F 是 T -不可近似的, 则 F' 是 T' -不可近似的, 其中 $T'(m + t'(m)) = \min(T(m) - O(m), \exp(t'(m)/O(1)))^{1/3}$ 。

提示: 将 F 的近似归约为 F' 的近似, 利用一个事实, 即对任意 $x = (x_1, x_2, \dots, x_{t(n)})$ 及 $r = r_1 r_2 \cdots r_{t(n)}$, 有 $\bigoplus_{i \in S_r} f(x_i) = F'(x, r)$, 其中 $S_r = \{i \in [t(n)] : r_i = 1\}$ 。注意: 集合 S_r 中包含至少 $t(n)/3$ 个下标的概率是 $1 - \exp(-\Omega(t(n)))$ 。因此, f 的 $t(n)/3$ 个取值的异或可以归约为 $t(n)$ 个这种值的选择性异或 (利用习题 7.17 的部分思想来处理 $|S_r| > t(n)/3$ 的情形)。 $t(n)$ 个值的异或可由 3 个异或 (每个含 $t(n)/3$ 个值) 得出, 这样做的代价是将优势变为 3 的幂, 从而造成优势减少。

7.19 (将“选择性异或”归约为直积) 回顾在 7.2.1.2 节中, 对“选择性异或”谓词 P'

^① 也就是说, 首先假设归约中给定了分布 $(U_n, f(U_n))$ 的 $t(n)$ 个样本, 并对一个均匀分布的输入 $(x, r) = (x_1, x_2, \dots, x_{t(n)}, r_1, r_2, \dots, r_{t(n)})$, 分析其成功概率。然后, 应用习题 7.16, 其中 X 表示实际输入 (x, r) 的分布, 而 Z 表示辅助样本序列的输入。

求近似被归约为对直积 P 取值的猜测。构造一个反向归约。即对如 7.2.1.2 节中的 P 和 P' ，证明如果 P' 是 T' -不可近似的，则任意 T -规模电路计算 P 的成功概率至多为 $1/T$ ，其中 $T = \Omega(T')$ 。

提示：利用习题 7.15。

7.20 (定理 7.14 与定理 7.15) 考虑定理 7.14 的一种推广，其中 f 和 P 是从串到串集的函数，满足 $P(x_1, x_2, \dots, x_t) = f(x_1) \times \dots \times f(x_t)$ 。

(1) 证明如果对任意 p_1 -规模电路类 $\{C_n\}_{n \in \mathbb{N}}$ ，及所有足够大的 $n \in \mathbb{N}$ ，有 $\Pr[C_n(U_n) \neq f(U_n)] > 1/p_2(n)$ ，则对任意 p' -规模的电路类 $\{C'_m\}_{m \in \mathbb{N}}$ ，有 $\Pr[C'_m(U_m) \in P(U_m)] < \varepsilon'(m)$ ，其中 ε' 和 p' 如定理 7.14 所述。进一步推广这个断言，将 $\{U_n\}_{n \in \mathbb{N}}$ 替换为任意一个概率空间 $\{X_n\}_{n \in \mathbb{N}}$ ，并将 U_m 用 X_n 的 $t(n)$ 倍笛卡儿积代替（其中 $m = t(n) \cdot n$ ）。

(2) 证明上述方法既是定理 7.14 的推广也是定理 7.5 的非一致复杂性版本。

7.21 (对 7.2.1.3 节主旨的提炼) 考虑对定义 7.17 的如下修改，其中译码条件针对着门限为 $(1/q(N)) + \alpha(N)$ 的协商，而非门限 $\alpha(N)$ 。修改后的定义如下（其中 p 为固定的多项式）：对任意 $\omega: [l(N)] \rightarrow [q(N)]$ 及 $x \in \{0, 1\}^N$ ，满足 $\Gamma(x)$ 与 ω 是 $(1 - (1/q(N)) + \alpha(N))$ -接近的，则存在一个有预言帮助的，规模为 $p((\log N)/\alpha(N))$ 的电路 C ，使得对任意 $i \in [N]$ ， $C^\omega(i)$ 为 x 的第 i 比特。

(1) 构造并证明定理 7.18 关于修改后定义的版本（不同于原始定义）。

提示：修改后的版本应涉及到以大于 $(1/q(N)) + \varepsilon(N)$ 的成功概率计算 $g(U_{m(n)})$ 。

(2) 证明当应用于二代码（即 $q \equiv 2$ ）时，(1) 中的版本可得到一个 S'' -不可近似谓词，其中 $S''(n') = S(m^{-1}(n'))^{\Omega(1)} / \text{poly}(n')$ 。

(3) 证明 Hadamard 乘积在修改后的定义中允许绝对译码（但在原始定义中不允许）。^①

提示：这是定理 7.8 的本质内容。

证明如果 $\Gamma: \{0, 1\}^N \rightarrow [q(N)]^{l(N)}$ 是一个允许绝对译码的（非二元）码，则对其符号进行 Hadamard 编码可以得到一个允许绝对译码的二代码（ $\{0, 1\}^N \rightarrow \{0, 1\}^{l(N) \cdot 2^{\lceil \log_2 q(N) \rceil}}$ ）。注意：仅当 $q(N) \leq \text{poly}(N)$ 时才保持编码的高效性。

7.22 (利用命题 7.16 证明定理 7.19) 结合命题 7.16 与定理 7.8 来证明定理 7.19。

提示：注意：对某个 $\gamma > 0$ ，命题 7.16 给出一个指数时间可计算的函数 $\hat{f}$ ，满足 $|\hat{f}(x)| \leq |x|$ ，且对任意规模为 $S'(n') = S(n'/3)^\gamma / \text{poly}(n')$ 的电路簇 $\{C'_{n'}\}_{n' \in \mathbb{N}}$ ，有 $\Pr[C'_{n'}(U_{n'}) = \hat{f}(U_{n'})] < 1/S'(n')$ 。结合定理 7.8，证明 $P(x, r) = b(\hat{f}(x), r)$ 是 S'' -不可近似的，其中 $|r| = |\hat{f}(x)| \leq |x|$ ， $S''(n'') = S(n''/2)^{\Omega(1)} / \text{poly}(n'')$ 。注意：如果 $S(n) = 2^{\Omega(n)}$ ，则 $S''(n'') = 2^{\Omega(n')}$ 。

7.23 令 G 为一个两两独立发生器（如引理 7.22 所述）， $S \subset \{0, 1\}^n$ 且 $\alpha \stackrel{\text{def}}{=} |S|/2^n$ 。证明 $G(U_{2n})$ 输出的 n 个串中至少有一个属于 S 的概率不小于 $\min(n \cdot \alpha, 1)/2$ 。进一步地，如果

^① 当然，Hadamard 编码并非高效编码（由于一个显然原因，即其码字具有指数长度）。

$\alpha \leq 1/2n$, 则这个概率至少为 $0.75 \cdot n \cdot \alpha$ 。

提示: 利用 G 的两两独立性质, 并根据容斥原理可求出上述概率的下限为 $n \cdot \alpha - \binom{n}{2} \cdot \alpha^2$ 。

如果 $\alpha \leq 1/n$, 则断言成立; 否则, 可对 $G(U_{2n})$ 输出的前 $1/\alpha$ 个元素应用同样的推理。

7.24 (单向函数与不可近似谓词) 证明非一致困难单向函数 (如定义 7.3) 的存在性蕴含了指数时间计算的、 T -不可近似谓词的存在性 (如定义 7.9), 其中 T 为任意多项式。

提示: 首先假设单向函数 f 是保持长度的双射。考虑由定理 7.8 保证的、 $g(x, r) = (f(x), r)$ 的困难核心谓词 b , 定义布尔函数 h , 满足 $h(z) = b(g^{-1}(z))$, 并证明对任意多项式 T , h 是 T -不可近似的。对于一般情形, 需要采用不同方法。具体地, 给定一个 (保持长度的) 单向函数 f , 考虑如下定义的布尔函数 h : 当且仅当 $f^{-1}(z) = \{x: f(x) = z\}$ 的字典序排列中的第 i 比特等于 σ 时, 令 $h(z, i, \sigma) = 1$ (特别地, 如果 $f^{-1}(z) = \emptyset$, 则对任意 i 及 σ , $h(z, i, \sigma) = 0$)。^① 注意: h 可在指数时间内计算, 但是 (在最坏情况下) 不能由多项式规模电路计算。应用定理 7.10, 命题得证。

^① 因此, h 在平均情况下可能是易计算的 (如对某个单向函数 f' , $f(x) = 0^{|x|} f'(x)$)。

第8章

伪随机数发生器

不可区分的事物是相同的。^①

戈特弗里德·威廉·莱布尼兹（1646—1714）

计算理论从一种新的角度看待随机性：如果无法使用高效的算法将一种分布与均匀分布区分开，则认为该分布是随机的（或伪随机的）。因此，（伪）随机性并不是研究目标的固有属性，而更像是观察者的主观感受。

极端情况下，这种观点认为，世界是确定的还是允许某种自由选择（可被视为随机源），这个问题并不重要。重要的是，在人类以及各种计算受限设备的眼中世界是什么样子，也就是说，如果某种现象看上去是随机的，则认为它就是随机的。类似地，如果可以生成一些序列，不能用高效算法区分其与均匀分布，则可将这些序列用于需要随机序列的高效应用中，而不是使用该应用规定的理想随机数。

上述方法的关键在于计算不可区分性的概念，它指不能由高效程序来区分一对分布。这一概念最基本的方面是将高效程序与多项式时间算法相关联，而其他方面则关注其他类型的区分程序，这也能得到重要结论。多数人关心如何高效生成具有伪随机性的对象，因为许多场合需要使用这些对象。

内容提要

伪随机数发生器是一些高效的确定算法，能将短的随机数种子扩展为更长的伪随机序列。其定义中包含 3 个具体内容——发生器的扩张方式、发生器要欺骗的区分器类型（即满足计算不可区分性要求的算法），以及发生器能使用的计算资源（即其自身的计算复杂度）。

伪随机数发生器的典型例子是可以欺骗任意可行程序的高效发生器，此时，区分器为任意一种概率多项式时间算法，它可能比发生器更复杂（反过来发生器的时间复杂度上限由一个固定多项式限定）。这类发生器被称为通用的，因为其输出可以安全地用于任意一种高效应用。这类（通用的）伪随机数发生器存在当且仅当单向函数存在。

与这类（通用的）伪随机数发生器相反，为了达到去随机化的目的，可以使用非严格定义的伪随机数发生器。特别地，此时只需使用比区分器（代表着将被去随机化的随机算法）更复杂的伪随机数发生器即可。利用这种方法，伪随机数发生器可对 BPP 类完全地去随机化（即 $BPP=P$ ）。构造此类发生器时，需要利用 ϵ 中存在一些不能用亚指数规模电路求解的问

^① 这是莱布尼兹的“不可区分的同一性原则”（Principle of Identity of Indiscernibles）。莱布尼兹认为，与这个定律相矛盾的例子有可能存在，但在现实中不会发生，因为上帝是如此仁慈。因此，我们相信他一定会赞同本章的主题，其中坚持将不可区分的事物看成是相同的。

题这一假设。

考虑可欺骗空间受限区分器的伪随机数发生器以及随机性受限的伪随机数发生器（如要求输出两两独立的序列或有小偏移量的序列）也很有意义。构造这些（特殊用途）的伪随机数发生器时，可以不依赖于计算复杂性假设。

引言

“随机性问题”已经困扰了人们数十年。该问题涵盖了从对（世界上）是否存在随机性的哲学思考到随机性的意义（在人们头脑中），以及随机性如何度量的技术问题。20 世纪后 50 年中，人们见证了 3 种随机性理论的发展，分别对应着随机性的上述 3 个方面。

第一种理论（见参考文献[63]）由 Shannon 提出^[204]，认为随机性代表信息的缺失，这又可以用缺失数据可能取值的概率分布来表示。事实上，Shannon 的信息论根植于概率论基础之上。信息论主要考虑并非完全随机的分布（即以冗余方式对信息编码），并将完全随机性视为一种极端情况，此时信息量达到最大（即根本没有任何冗余）。因此，完全随机性只涉及一种分布——均匀分布。特别地，根据定义，无法由较短的随机种子（确定地）生成完全随机的串。

第二种理论（见参考文献[153, 156]），由 Solomonoff^[210]、Kolmogorov^[147]和 Chaitin^[51]提出，认为随机性代表结构上的缺失，这又体现于对一个对象最简洁和高效的描述长度中。简洁和高效的描述涉及到一个过程，能将简洁描述转化为明确描述。事实上，随机性的这种理论根植于可计算理论，特别是在通用语言的意义上（等价于通用机或计算设备，见 1.2.3.4 节）。它用（在固定的通用机上）生成一个对象的最短程序来衡量对象的随机性（或复杂性）^①。与 Shannon 理论相似，柯尔莫果洛夫复杂性是量化的，将完全随机的对象作为一种极端情形。然而，在这种理论中，可以称一个独立的对象，而非对象的一种分布，是完全随机的。根据定义，仍不能（确定地）由短的随机数种子生成具有较高柯尔莫果洛夫复杂度的串。

第三种理论是本章的重点，认为随机性是观察者的一种印象，它与观察者的（分析）能力有关。观察者的能力主要表现为计算能力（即观察者可以应用的程序的复杂性）。因此，随机性的这种理论根植于复杂性理论基础上。其中一个明确目标就是给出不同于前两种理论的随机性定义，并允许由短的随机种子高效地（且确定地）生成随机串。这种理论的核心是，如果不能由高效过程区分两个对象，则认为两者相同。从而，如果无法高效地区分一种概率分布与均匀分布，则视其为随机的（或更确切地称其为伪随机的）。因此，随机性不是对象（或分布）的固有属性，而是一种与观察者（及其计算能力）相关的属性。为阐述这种方法，考虑如下的智力游戏。

Alice 和 Bob 以下列四种方式之一玩掷硬币游戏。每次游戏中，Alice 掷一枚公平的硬币，在着地之前让 Bob 猜结果。根据 Bob 在猜测前拥有的知识不同，可将游戏方式分为四种。

第一种方式，在 Alice 掷硬币之前，Bob 必须宣布其猜测，显然，此时 Bob 猜对的概率为 1/2。

第二种方式，当硬币在空中旋转时，Bob 必须宣布其猜测。虽然此时原则上掷硬币的结果取决于硬币的运动规律，但 Bob 对此并不清楚，因此我们相信这种情况下，Bob 猜对的概率仍为 1/2。

第三种方式与第二种相似，但此时 Bob 可以使用一种复杂设备来得到硬币行为的准确信

① 我们指出 Kolmogorov 方法本质上是很难的（即 Kolmogorov 复杂度是不可计算的）。

息，以及对结果有影响的环境信息。然而，Bob 不能及时利用这些信息来纠正其猜测。

第四种方式，Bob 使用的设备直接连到强大的计算机上，该计算机能解运动方程，并输出预测。可以想像，此时 Bob 一定会利用机器的输出来纠正其猜测。

总结一下，事件的随机性与可以使用的信息和计算资源有关。在极端情况下，如果观察者缺乏相关信息或处理信息的能力，则可认为由公开信息完全确定的平凡事件也是随机的。我们重点讨论缺乏处理能力的情形，它可能由（用于分析事件的）计算资源不够造成，也可能由观察者的能力受限造成。

这就自然地产生了伪随机性的概念——称一种分布是伪随机的，如果不存在高效程序能区分其与均匀分布。这里的高效程序与（概率）多项式时间算法有关。实际上，这是伪随机性最基本的定义，本章将用大部分篇幅讨论。还有一种更弱的伪随机性概念——不能由较弱的程序（如空间受限算法，常数深度电路等等）区分。对这些算法进一步扩展，可以考虑有意设计一些算法，使其甚至不能区分更弱的“伪随机”序列与真正的随机序列（当我们要将某些自然的随机算法转化为确定算法时，就需要构造这些算法，见 8.5 节）（图 8.1）。

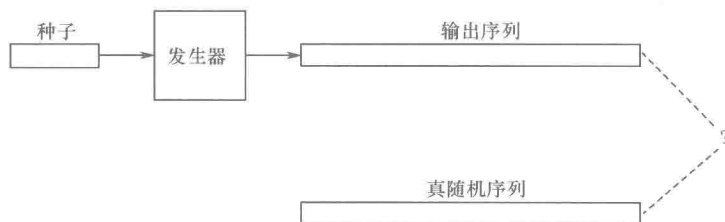


图 8.1 伪随机性发生器——图示

上述讨论集中于伪随机性的一个方面——资源或观察者（或潜在区分器）类型。另一个重要方面是这些伪随机序列是否能由更短的序列产生，以及生成这些序列的代价（或复杂性）。当然，要求产生过程是高效的，进一步地，这些过程在特定观察者做出判断之前还应是固定的。从上述的强伪随机性概念中，可以得到伪随机数发生器的典型概念——运行于（固定的）多项式时间，且任意一个多项式时间观察者不能区分生成的序列与均匀序列。特别地，这意味着区分器比发生器要使用更多资源。这些（通用的）伪随机数发生器（见 8.2 节）能降低高效应用程序的随机性复杂度，因此与随机算法和密码学关系紧密。使用“通用”的术语是为了强调同一发生器适用于所有的高效应用，包括那些比发生器本身要消耗更多资源的应用。

虽然通用的伪随机数发生器用途很广，但发生器与区分器复杂度间的另一种相反关系也不容忽视，即伪随机数发生器比其要欺骗的观察者使用更多（时间或空间）资源。在去随机化（将随机算法转化为确定算法）中，这很正常，因为其中的关键步骤是将算法的随机输入用伪随机输入代替，后者则由一个更短的随机数种子产生。特别地，在对概率多项式时间算法去随机化时，（发生器要欺骗的）观察者是一个固定算法。此时使用更复杂的发生器仅意味着得到的确定算法的复杂度取决于发生器的复杂度（而非原始随机算法的复杂度）。当然，在允许发生器比其要欺骗的观察者使用更多资源时，伪随机数发生器的设计将更容易，而且得到的去随机化结果是利用通用的伪随机数发生器所无法得到的。8.3 节~8.5 节阐述这种方法的用途。

注意：所有伪随机数发生器的功能都是基于短的随机数种子来生成“足够随机”的序列。因此，伪随机数发生器可以显著减小各种应用的随机性复杂度（在某些情况下甚至完全去除随机性）。随机性的减少很有意义，因为许多应用程序生成或得到真随机比特序列的能力极为有限。另外，典型地，生成真随机比特序列远比标准的计算步骤花费代价大得多。因此，除

时间之外, 随机性也是一种要考虑的计算资源 (类似于空间)。

内容组织

8.1 节给出伪随机数发生器的一般概念。8.2 节给出通用伪随机数发生器的典型实例。然后讨论伪随机数发生器的其他概念: 8.3 节讨论用于对复杂性类如 BPP 类去随机化的发生器; 8.4 节讨论空间受限计算中的伪随机数发生器; 8.5 节讨论特殊用途的发生器。

教学注释: 如果只能讲述一种伪随机数发生器, 则我们建议选择通用的伪随机数发生器 (见 8.2 节)。它与计算机科学中多个领域紧密相关, 并且技术细节较为简单。本章的内容组织也是为了方便这种选择。

基础知识

假设读者熟悉概率论基础 (见附录 D.1) 以及随机算法 (6.1 节)。特别地, 本章大量使用随机变量的知识和习惯用法 (见附录 D.1.1)。本章还自包含地选用了第 7 章的一些结论。

8.1 通用模式

教学注释: 我们提炼出伪随机数发生器各种定义中的共同点。即把这些概念看作是一种通用抽象模型的不同实现。如果要重点讲述其中之一, 可利用本节来引入后面的具体定义。另一方面, 一些学生可能更愿意在学习了一种具体实现之后, 再回过头来学习本节。

伪随机数发生器的一般定义包含 3 个具体方面——扩张方式、区分器类型以及使用的计算资源 (即其计算复杂度)。

扩张函数

伪随机数发生器的必要条件是必须为一个确定性算法, 能将较短的、称为种子的串扩张为较长的输出序列。^①具体而言, 算法将 k 比特种子扩张为 $l(k)$ 比特输出, 其中 $l(k) > k$ 。函数 $l: \mathbf{N} \rightarrow \mathbf{N}$ 称为扩张方式 (或扩张函数)。在某些环境中, 具体的扩张方式并不重要 (如可见 8.2.4 节)。

计算不可区分性

伪随机数发生器的另一个必要条件是它能“骗过”某个非平凡算法。即从预先指定的算法集合中任意选择的算法不能区分发生器的输出 (当输入为均匀选择的种子时) 与均匀选择的序列。典型地, 考虑 $\mathcal{D}$ 类区分器 (概率多项式时间算法) 以及 $\mathcal{F}$ 类 (限门) 函数, 并要求发生器 G 满足下述条件: 对任意 $D \in \mathcal{D}$, $f \in \mathcal{F}$, 以及所有足够大的 k , 有

$$\left| \Pr[D(G(U_k))=1] - \Pr[D(U_{l(k)})=1] \right| < f(k) \quad (8.1)$$

其中 U_n 表示 $\{0,1\}^n$ 上的均匀分布, 其概率空间为在 U_k (或 $U_{l(k)}$) 以及算法 D 的内部随机数。读者也许会这样的区分器 D 视为一个观察者, 试图判别“被测试串”是发生器的随机输出 (服从分布 $G(U_k)$) 还是一个真正的随机串 (服从分布 $U_{l(k)}$)。式 (8.1) 中的条件要求 D 无法做出有意义的判定, 即忽略一个微小差别 (表示为 $f(k)$) 时, D 在两种情况下将做出相同

^① 事实上, 种子代表为生成输出序列所需的随机性, 即把随机化的生成过程弱化为一个附带了随机种子的确定算法。这种弱化便于研究随机数的产生过程。

的判定。^①一个典型选择是令 $\mathcal{D}$ 为所有概率多项式时间算法, $\mathcal{F}$ 是可作为某个正多项式倒数的所有函数集合。

生成随机数的复杂度

照例要求发生器工作于多项式时间内(是其输入(即种子)长度的多项式)。我们还将考虑别的要求。注意:如果对于发生器没有任何计算上的要求(或者要求比较温和,如运行时间的上限为双指数(Double-exponential)函数),则构造的随机数发生器可以骗过任何亚指数规模的电路类(见习题 8.1)。

常用符号

用 k 表示伪随机数发生器的种子长度, $l(k)$ 表示相应输出的长度。在某些情况下,这样表示会带来一些麻烦(因为更自然的方法是规定一些参数,并令种子长度为这些参数的函数)。然而,这样做最大的好处是可以集中注意于伪随机数发生器的基本参数——即随机种子的长度。注意:无论何时,在使用伪随机数发生器对某个算法“去随机化”时,都用 n 表示该算法的输入长度,并令 k 为 n 的函数。

通用模式的几个实例

伪随机数发生器的两个重要实例与概率多项式时间区分器有关。

(1) 通用的伪随机数发生器对应于这样一种情况:发生器本身运行于多项式时间,并要经受任意概率多项式时间区分器的攻击,包括那些运行时间多于发生器的区分器。因此,同一发生器可以安全地用于任意高效的应用中(8.2 节将详细介绍)。

(2) 相反,用于去随机化的伪随机数发生器其运行时间可以多于区分器,可将其视为固定的电路,其规模上限为某个固定的多项式。(这一概念在 8.3 节详细介绍。)

另外,通用模式在实例化时,还可关注区分器(以及发生器)的空间复杂度,而非时间复杂度。进一步地,可以考虑那些仅符合某些概率性质的区分器,如两两独立、小偏移量以及命中频率的区分器。

8.2 通用的伪随机数发生器

随机性在计算中起着越来越重要的作用,常用于设计串行、并行及分布式算法,并且理所当然地在密码学中占据了核心地位。虽然设计一个能随意使用随机数的算法很容易,但在真正实现时,却最好将使用的随机数个数减至最少。因此,通用的伪随机数发生器(随后给出定义)在“算法工具箱”中是一个关键元素——它们将一个可以随意使用随机数的算法自动编译为“经济”地使用随机数的程序。

本节内容组织

本节相对较长,所以有必要提供一个简单的阅读指南。8.2.1 节给出通用伪随机数发生器的定义,8.2.2 节描述其典型应用(前面的内容都未涉及)。8.2.3 节在计算不可区分性的基本定义之外给出更具外延性的讨论。8.2.4 节解释特殊的扩张函数(见 8.2.1 节)。8.2.5 节讨论通用伪随机数发生器的存在性。8.2.6 节讨论计算不可区分性的非一致版本。最后在 8.2.7 节进行总结,回顾并思考本节的各种概念和定义。

^① 限门函数类 $\mathcal{F}$ 可以看作是判定具有明显概率(作为 k 的函数)的类。因此,要求特定函数(即式(8.1)中的概率差)除对于有限个整数外,应小于任意不可忽略的函数。称前一个函数是可忽略的。注意:一个函数可以既非显著的也非可忽略的(如它可以在无穷多个点小于任意显著的函数,但是也在无穷多个其他点大于某些显著的函数)。

8.2.1 基本概念

不严格地说, 通用伪随机数发生器是一些高效的(即多项式时间)的确定算法, 可将短的随机数种子扩张为更长的伪随机比特序列, 后者与真正的随机序列利用任意高效算法(即多项式时间)在计算上不可区分。高效性用多项式时间表示, 这意味着(作为固定算法的)发生器运行于某个固定的多项式时间, 而区分器可以是运行于任意多项式时间的算法。因此, 区分器比发生器复杂度更高。例如, 区分器的运行时间为发生器运行时间的3次方。进一步地, 为了便于理论上的扩展, 允许区分器为概率算法(而发生器仍为确定算法)。要求这种区分器不能区分发生器的输出与等长的真随机序列。或者说, 区分器能检测出两者区别的概率可以忽略不计。这里可忽略的函数指比任意多项式的倒数减少得更快的函数。^①

定义 8.1 (通用的伪随机数发生器) 称确定的多项式时间算法 G 为伪随机数发生器, 如果存在一个扩张函数 $l: \mathbf{N} \rightarrow \mathbf{N}$ (满足对任意 k , 有 $l(k) > k$), 使得对任意概率多项式时间算法 D , 任意正多项式 p , 以及足够大的 k , 有

$$\left| \Pr[D(G(U_k))=1] - \Pr[D(U_{l(k)})=1] \right| < \frac{1}{p(k)} \quad (8.2)$$

其中 U_n 表示 $\{0,1\}^n$ 上的均匀分布, 概率空间为 U_k (或 $U_{l(k)}$) 以及 D 的所有可能的内部随机数。

定义 8.1 可由 (8.1 中给出的) 通用框架得出, 只需令其中的区分器类为所有概率多项式时间算法, 并令 (显著的) 限门函数为可表示为某个正多项式倒数的函数。^②事实上, 定义 8.1 中的原则在 8.1 节已经讨论过 (并将在 8.2.3 小节中进一步讨论)。

注意: 定义 8.1 对于扩张函数 $l: \mathbf{N} \rightarrow \mathbf{N}$ 并未做任何要求, 除了一般性地要求对任意 k , $l(k) > k$ 之外。当然, l 越大, 伪随机数发生器就越有用。 l 的上限显然为发生器的运行时间 (因此, 是一个多项式)。8.2.4 节将证明任意伪随机数发生器 (即使具有最小扩张 $l(k) = k+1$) 都可用于构造具有某种期望 (多项式的) 扩张函数的伪随机数发生器。但在此之前需严谨地讨论由伪随机数发生器所导致的“随机性的减少”, 并对定义 8.1 中的计算不可区分性有更深入的理解。

8.2.2 典型应用

“伪随机数发生器”伴随着第一台计算机的出现而出现, 从那以后一直在各种应用中用于生成随机选项 (或采样)。然而, 发生器的典型实现并非如定义 8.1 中是伪随机的。相反, 这些发生器充其量是被证明了能通过某种统计测试 (见参考文献[146])。要注意的是, “伪随机数发生器”通过了某些统计测试并不意味着它也能通过一种新的测试, 并能满足某种未来的 (未被测试的) 应用。当然, 用某种专门测试来评估发生器性能, 这种方法并不能保证“在所有应用中使用该发生器相当于使用真正公平的掷硬币结果”。相反, 定义 8.1 中的方法则针对着这种一般性, 并试图达成这个目标: 定义 8.1 中强调的计算不可区分性, 包含了所有可能

① 定义 8.1 要求表示某些算法区分缝隙的函数应该在除有限个 k 之外, 对几乎所有 k 的取值都小于任意正多项式的倒数。这些函数被称为可忽略的 (见脚注 4, 其中用任意正多项式的倒数识别显著函数)。可忽略的概率定义在下述意义上是很强的: 任意以可忽略概率发生的事件, 当实验重复进行“可行的” (即多项式) 次数时, 该事件发生的概率仍可忽略。

② 后者常伴随着多项式时间算法的高效计算: 以显著概率发生的事件, 当实验重复进行“可行的” (即多项式) 次数时, 该事件几乎总是会发生。

的高效应用，并在这些应用中确保伪随机序列与真随机序列起着相同作用。事实上，在任意高效的随机算法中，当用伪随机数发生器的输出代替算法的内部随机数时，都能保持其性能。下面解释这种代替。

构造 8.2 (伪随机数发生器的典型应用) 设 G 为伪随机数发生器，扩张函数为 $l: \mathbf{N} \rightarrow \mathbf{N}$ 。设 A 为概率多项式时间算法， $\rho: \mathbf{N} \rightarrow \mathbf{N}$ 表示其随机性复杂度。将 A 对于输入 x 以及随机输入序列 $r \in \{0,1\}^{\rho(|x|)}$ 的输出记作 $A(x,r)$ 。考虑如下的随机算法 A_G 。

对于输入 x ，令 $k = k(|x|)$ 为满足 $l(k) \geq \rho(|x|)$ 的最小正整数，均匀选择 $s \in \{0,1\}^k$ ，并输出 $A(x,r)$ ，其中 r 为 $G(s)$ 的 $\rho(|x|)$ -比特前缀。

也就是说，对 $|s| = k(|x|) = \arg \min_i \{l(i) \geq \rho(|x|)\}$ ，有 $A_G(x,s) = A(x, G'(s))$ ，其中 $G'(s)$ 为 $G(s)$ 的 $\rho(|x|)$ -比特前缀。

因此，利用算法 A_G 来代替 A ，可将随机性复杂度由 ρ 降低为 $l^{-1} \circ \rho$ 。然而 (以下将证明)，不可能找到一些输入 (即 x) 使得 A_G 在计算 x 时与 A 的行为明显不同。例如，如果 $l(k) = k^2$ ，则随机性复杂度由 ρ 降低为 $\sqrt{\rho}$ 。我们强调伪随机数发生器 G 是通用的，即可以用于降低任意概率多项式时间算法 A 的随机性复杂度。

命题 8.3 设 A 、 ρ 和 G 如构造 8.2，并假设 $\rho: \mathbf{N} \rightarrow \mathbf{N}$ 为双射，则对任意一对概率多项式时间算法，即一个搜索器 F 和检测器 T ，任意正多项式 p 和足够长的 n ，有

$$\sum_{x \in \{0,1\}^n} \Pr[F(1^n) = x] \cdot |\Delta_{A,T}(x)| < \frac{1}{p(n)} \quad (8.3)$$

其中

$$\Delta_{A,T}(x) \stackrel{\text{def}}{=} \Pr\left[T\left(x, A\left(x, U_{\rho(|x|)}\right)\right) = 1\right] - \Pr\left[T\left(x, A_G\left(x, U_{k(|x|)}\right)\right) = 1\right]$$

算法 F 尝试寻找这样的 x ，使得 A_G 对于 x 的输出与 A 的输出不可区分。当用户使用算法 A_G ，并且其输入由某个概率多项式时间算法产生时，这种“尝试”是有用的。然而，当算法 A_G 的输入由某个恶意参与方产生时，则这种尝试也是有恶意的。潜在的检测器，记作 T ，表示暗中使用算法 A_G 的输出时，要求该输出与 A 的输出一样“好”。因此， T 的输入为 x ，以及或由 $A_G(x) \stackrel{\text{def}}{=} A(x, U_{k(|x|)})$ 或由 $A(x) = A(x, U_{\rho(|x|)})$ 产生的输出，并要求 T 无法区分这两者。当 A 为概率多项式时间的判定程序时，无法区分意味着不可能找到这样的 x ，使 A_G 作出误判 (即不同于 A)。当 A 为某个 NP -关系的搜索程序时，意味着不可能找到 x ，使 A_G 输出错误的解。细节详见习题 8.2。

证明：要证明该命题，需要证明任意一个不符合命题的三元组 (A, F, T) 可被转化为算法 D ，能区分 G 的输出与均匀分布，从而与假设矛盾。重要的是，对任意 $x \in \{0,1\}^n$ ，有

$$\Delta_{A,T}(x) = \Pr\left[T\left(x, A\left(x, U_{\rho(n)}\right)\right) = 1\right] - \Pr\left[T\left(x, A\left(x, G'\left(U_{k(n)}\right)\right)\right) = 1\right] \quad (8.4)$$

其中 $G'(s)$ 为 $G(s)$ 的 $\rho(n)$ -比特前缀。因此，求使 $|\Delta_{A,T}(x)|$ 较大的串 x 的算法可以用于构造区分 $U_{l(k(n))}$ 与 $G(U_{k(n)})$ 的方法。即给定样本 $r \in \{0,1\}^{l(k(n))}$ ，利用满足条件的串 x ，区分器的输出为 $T(x, A(x, r'))$ ，其中 r' 为 r 的 $\rho(n)$ -比特前缀。实际上是要证明，当式 (8.3) 不成立时，这

针对着 $E_{x \leftarrow F(1^n)}[\Delta_{A,T}(x)]$, 将与 G 为伪随机数发生器的假设矛盾 (通过找到并使用足够多的 x 可证明这一点)。这种直观的方法需要更加谨慎地实现, 以下给出。

作为热身, 考虑如下算法 D : 对于输入 r (来自于 $U_{l(k(n))}$ 或 $G(U_{k(n)})$), 算法 D 首先得到 $x \leftarrow F(1^n)$, 其中 n 易由 $|r|$ 得出 (因为 ρ 是双射, 可通过 A 计算 $1^n \mapsto \rho(n)$)。下一步, D 得到 $y = A(x, r')$, 其中 r' 为 r 的 $\rho(|x|)$ -比特前缀。最后, D 输出 $T(x, y)$ 。注意: D 可在概率多项式时间内实现, 并且

$$D(U_{l(k(n))}) \equiv T(X_n, A(X_n, U_{\rho(n)})), \text{ 其中 } X_n \stackrel{\text{def}}{=} F(1^n)$$

$$D(G(U_{k(n)})) \equiv T(X_n, A(X_n, G'(U_{k(n)}))), \text{ 其中 } X_n \stackrel{\text{def}}{=} F(1^n)$$

根据式 (8.4), $\Pr[D(U_{l(k(n))})=1] - \Pr[D(G(U_{k(n)}))=1]$ 与 $E[\Delta_{A,T}(F(1^n))]$ 相等, 这意味着 $E[\Delta_{A,T}(F(1^n))]$ 一定是可忽略的 (否则, 就与 G 是伪随机数发生器的假设矛盾)。这就得到了命题的一个较弱版本, 即 “ $E[\Delta_{A,T}(F(1^n))]$ 可忽略” (并非 $E[\Delta_{A,T}(F(1^n))]$ 可忽略)。

为证明 $E[\Delta_{A,T}(F(1^n))]$ (而非 $E[\Delta_{A,T}(F(1^n))]$) 是可忽略的, 需要对 D 稍做修改。注意: 麻烦之处在于 $\Delta_{A,T}(\cdot)$ 可能对某些 x 为正而对另一些为负, 因此有可能当 $E[\Delta_{A,T}(F(1^n))]$ 较大时, $E[\Delta_{A,T}(F(1^n))]$ 却较小。为解决这一困难, 可对 $x = F(1^n)$ 判定 $\Delta_{A,T}(\cdot)$ 的符号, 并据此修改 D 的输出。换言之, 修改后的 D 当 $\Delta_{A,T}(x) > 0$ 时输出 $T(x, A(x, r'))$, 否则, 输出 $1 - T(x, A(x, r'))$ 。从而, 在任何情况下, x 使修改后的 D 的区分缝隙增加 $|\Delta_{A,T}(x)|$ 。进一步地, 还要注意, 如果 $|\Delta_{A,T}(x)|$ 较小, 则无论 $\Delta_{A,T}(x) > 0$ 还是 $\Delta_{A,T}(x) \leq 0$ 都无关紧要。因此, 只需在 $|\Delta_{A,T}(x)|$ 较大时, 正确判定 $\Delta_{A,T}(x)$ 的符号即可, 这显然是一个 (近似) 可行的任务。具体细节如下。

用反证法, 假设对某个不可忽略的函数 ε , 有 $E[\Delta_{A,T}(F(1^n))] > \varepsilon(n)$, 对于输入 r (服从分布 $U_{l(k(n))}$ 或 $G(U_{k(n)})$), 修改后的算法 D 首先得到 $x \leftarrow F(1^n)$, 这与基本版本相同。然后利用规模为 $\text{poly}(n/\varepsilon(n))$ 的采样估算 $p_U(x) \stackrel{\text{def}}{=} \Pr[T(x, A(x, U_{\rho(n)}))=1]$ 以及 $p_G(x) \stackrel{\text{def}}{=} \Pr[T(x, A(x, G'(U_{k(n)})))=1]$, 使得对每个概率的估值误差在 $\varepsilon(n)/8$ 以内的概率可忽略 (如 $\exp(-n)$) (注意到, 目前为止, D 的行为只依赖于输入 r 的长度, 后者又决定了 n)^①。如果这些近似值表明 $p_U(x) \geq p_G(x)$ (等价地, 表明 $\Delta_{A,T}(x) \geq 0$), 则 D 输出 $T(x, A(x, r'))$, 否则, 输出 $1 - T(x, A(x, r'))$, 其中 r' 为 r 的 $\rho(|x|)$ -比特前缀, 并且不失一般性, 设 T 的输出属于 $\{0, 1\}$ 。

① 具体地, 在估算 $p_U(x)$ (或 $p_G(x)$) 时, 先生成 $U_{\rho(n)}$ (或 $G'(U_{k(n)})$) 的一个样本, 再调用算法 A 和 T , 即给定 $U_{\rho(n)}$ (或 $G'(U_{k(n)})$) 的样本 $r_1, r_2, \dots, r_t$, 其中 $t = O(n/\varepsilon(n)^2)$, 我们根据 $\left| \{i \in [t] : T(x, A(x, r_i)) = 1\} \right| / t$ 来估算 $p_U(x)$ (或 $p_G(x)$)。

对修改后的区分器 D 的分析是基于一个事实：如果估算能得到关于 $p_U(x)$ 和 $p_G(x)$ 之间关系的正确判定，则 x 对 D 的区分缝隙的贡献为 $|p_U(x) - p_G(x)|$ 。^① 还要注意，如果 $|p_U(x) - p_G(x)| > \varepsilon(n)/4$ ，则“对 $p_U(x) - p_G(x)$ 的估值保持符号”以压倒性优势概率（即 $1 - \exp(-n)$ ）成立（因为两个不等式在求近似值时的加法错误都小于 $\varepsilon(n)/8$ ）。最后，注意：如果 $|p_U(x) - p_G(x)| \leq \varepsilon(n)/4$ ，则常常会产生 $p_U(x) - p_G(x)$ 的符号错误，因为由这个误差所导致的偏移（对于 D 的区分缝隙）最多为 $2|p_U(x) - p_G(x)| \leq \varepsilon(n)/2$ 。结合上述这些考虑，有

$$\begin{aligned} \Pr\left[D\left(U_{l(k(n))}\right)=1 \mid F\left(1^n\right)=x\right] - \Pr\left[D\left(G\left(U_{k(n)}\right)\right)=1 \mid F\left(1^n\right)=x\right] \\ \geq |p_U(x) - p_G(x)| - \eta(x) \end{aligned} \quad (8.5)$$

其中当 $|p_U(x) - p_G(x)| \leq \varepsilon(n)/4$ 时， $\eta(x) = \varepsilon(n)/2$ ，否则， $\eta(x) = \exp(-n)$ （事实上， $\eta(x)$ 表示由于误判 $p_U(x) - p_G(x)$ 的符号而造成的平均误差，其中当 $|p_U(x) - p_G(x)| \leq \varepsilon(n)/4$ 时，（由误判引起的）误差上限为 $\varepsilon(n)/2$ ，而当 $|p_U(x) - p_G(x)| > \varepsilon(n)/4$ 时， $\exp(-n)$ 为误差上限）。因此，式（8.5）的期望值就表示 $\Pr\left[D\left(U_{l(k(n))}\right)=1\right] - \Pr\left[D\left(G\left(U_{k(n)}\right)\right)=1\right]$ 的下限，等于 $E\left[\Delta_{A,T}\left(F\left(1^n\right)\right)\right] - E\left[\eta\left(F\left(1^n\right)\right)\right]$ 。结合 $E\left[\Delta_{A,T}\left(F\left(1^n\right)\right)\right] > \varepsilon(n)$ 的假设与 $\max_{x \in \{0,1\}^n} \{\eta(x)\} \leq \varepsilon(n)/2$ ，有 $\Pr\left[D\left(U_{l(k(n))}\right)=1\right] - \Pr\left[D\left(G\left(U_{k(n)}\right)\right)=1\right] > \varepsilon(n)/2$ 。回顾 D 运行于时间 $\text{poly}(n/\varepsilon(n))$ ，这与 G 的伪随机性相矛盾。命题得证。 ■

结论

虽然上文使用了标准的概率多项式时间算法，但类似的构造及分析可以用于任意一个高效的随机算法中（如高效的多方计算）。在这个过程中，（具体实现时）将真随机数用伪随机数替换，而仍保持其性质。因此，给定一个具有较大扩张函数的伪随机数发生器，可以极大地降低高效应用程序的随机性复杂度。

8.2.3 计算不可区分性

本节阐明（并研究）定义 8.1 中的计算不可区分性。

8.2.3.1 一般形式

计算不可区分性（的一般定义）针对着任意的概率空间。这里的概率空间是随机变量 $\{Z_n\}_{n \in \mathbb{N}}$ 的无限序列，每个 Z_n 表示长度是以 n 为变量的多项式的串（即存在多项式 p ，满足对任意 n ，有 $|Z_n| \leq p(n)$ 和 $p(|Z_n|) \geq n$ ）。称 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 计算不可区分，如果对任意可行的算法 A ，概率差 $d_A(n) \stackrel{\text{def}}{=} |\Pr[A(X_n)=1] - \Pr[A(Y_n)=1]|$ 是关于 n 的可忽略的函数。即有如下定义。

定义 8.4（计算不可区分性） 概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 是计算不可区分的，如果对任

① 事实上，如果 $p_U(x) \geq p_G(x)$ ，则该贡献为 $p_U(x) - p_G(x) = |p_U(x) - p_G(x)|$ ，而当 $p_U(x) < p_G(x)$ 时，则变成了 $(1 - p_U(x)) - (1 - p_G(x)) = -(p_U(x) - p_G(x))$ ，这也等于 $|p_U(x) - p_G(x)|$ 。

任意概率多项式时间算法 D , 任意正多项式 p , 及所有足够大的 n , 有

$$\left| \Pr[D(X_n)=1] - \Pr[D(Y_n)=1] \right| < \frac{1}{p(n)} \quad (8.6)$$

其中概率取值来自于相关分布 (即 X_n 或 Y_n) 以及算法 D 的内部随机数。式 (8.6) 左边被看作 n 的函数时, 常常称为 D 的区分缝隙, 其中 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 的含义见上文。

可以认为, D 是某个人, 想要区分两种分布 (基于取自某个分布的给定样本), 并认为输出 “1” 表示 D 判定该样本服从第一个分布。称两种分布在计算上不可区分, 意思是如果 D 是一个可行的程序, 则其判决是无意义的 (因为当样本来自第一种分布时输出 1 的频率与来自第二种分布时输出 1 的频率大体接近)。可以去掉式 (8.6) 中的绝对值符号而不影响定义的实质 (见习题 8.3), 下文即是如此。

定义 8.1 要求概率空间 $\{G(U_k)\}_{k \in \mathbb{N}}$ 和 $\{U_{l(k)}\}_{k \in \mathbb{N}}$ 是计算不可区分的。事实上, 定义 8.4 的一个重要特例是当其中一个空间为均匀分布时, 称另一个空间为伪随机的。

3.2.3.2 与统计接近的关系

称两个概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 统计接近 (或统计不可区分), 如果对任意正多项式 p 及所有足够大的 n , X_n 与 Y_n 的距离 (即 $\frac{1}{2} \sum_z |\Pr[X_n=z] - \Pr[Y_n=z]|$) 上限为 $1/p(n)$ 。显然, 任意两个统计接近的概率空间也是计算不可区分的。当然, 从信息论角度看, 统计接近只是计算不可区分的平凡情况。然而, 我们感兴趣的是 (计算不可区分性的) 非平凡情况, 对应于统计距离较远的概率空间。

事实上, 如 8.1 节所述, 确实存在统计距离较远而仍计算不可区分的概率空间 (见习题 8.1)。然而, 在习题 8.1 的概率空间中, 至少有一个无法在多项式时间内构造。^①我们对计算不可区分性的非平凡情况更感兴趣, 此时, 两个空间都可在多项式时间内构造。伪随机数发生器的定义 (见习题 8.7) 提供了一个重要例子。正如定理 8.11 将阐明, 单向函数的存在性蕴含了伪随机数发生器的存在性, 后者又蕴含了存在着多项式时间构造的概率空间, 统计距离较远但计算上不可区分。注意: 这个条件既是充分的, 也是必要的 (见习题 8.9)。

3.2.3.3 多样本不可区分性

计算不可区分性的定义 (即定义 8.4) 中涉及的区分器可以获得来自两个概率空间 (即 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$) 之一的样本。更一般地, 定义 8.4 可以自然地推广到能获得来自于概率空间之一的多个独立样本的区分器。

定义 8.5 (多个样本的不可区分性) 令 $s: \mathbb{N} \rightarrow \mathbb{N}$ 为多项式界函数。称两个概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 由 $s(\cdot)$ 个样本计算不可区分, 如果对任意概率多项式时间算法 D , 任意正多项式 $p(\cdot)$ 以及所有足够大的 n , 有

$$\left| \Pr[D(X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(s(n))})=1] - \Pr[D(Y_n^{(1)}, Y_n^{(2)}, \dots, Y_n^{(s(n))})=1] \right| < \frac{1}{p(n)}$$

其中 $X_n^{(1)}$ 到 $X_n^{(s(n))}$ 和 $Y_n^{(1)}$ 到 $Y_n^{(s(n))}$ 为独立随机变量, 且每个 $X_n^{(i)}$ 与 X_n 相同, 每个 $Y_n^{(i)}$ 与 Y_n 相同。

在一些最吸引人的情况下, 单个样本的计算不可区分性蕴含了任意多项式个样本的计算

① 称 $\{Z_n\}_{n \in \mathbb{N}}$ 可在多项式时间内构造, 如果存在一个多项式时间算法 S , 使得 $S(1^n)$ 和 Z_n 分布相同。

不可区分性。其中一种情况是可在多项式时间构造的概率空间。称概率空间 $\{Z_n\}_{n \in \mathbb{N}}$ 可在多项式时间内构造, 如果存在一个多项式时间算法 S , 使得 $S(1^n)$ 和 Z_n 分布相同。

命题 8.6 设 $X = \{X_n\}_{n \in \mathbb{N}}$ 和 $Y = \{Y_n\}_{n \in \mathbb{N}}$ 可在多项式时间内构造, s 为多项式, 则 X 与 Y 由单个样本计算不可区分当且仅当它们由 $s(\cdot)$ 个样本计算不可区分。

显然, 对任意多项式 s , 由 $s(\cdot)$ 个样本计算不可区分蕴含了由单个样本计算不可区分 (见习题 8.5)。现在证明对于可高效构造的概率空间, 单样本的不可区分性蕴含了多样本的不可区分性^①。以下证明简单地演示了一种核心证明技术, 称为混合 (Hybrid) 技术。

证明概要:^② 证明中再次利用可约性方法 (见前面的 7.2 节)。具体而言, 要证明如果存在一个高效算法, 能利用若干个样本区分 X 与 Y , 则也存在高效算法, 能利用单个样本区分 X 与 Y 。利用以下的“混合方法” (Hybrid Argument) 证明这一点。

为证明能够区分来自 X_n 的 $s(n)$ 个样本的序列与来自 Y_n 的 $s(n)$ 个独立样本序列, 考虑一个“混合”序列, 其中第 i 个元素包含 X_n 的 i 个样本, 继之以 $s(n) - i$ 个 Y_n 的样本。“同构”序列 (将证明是计算不可区分的序列) 是“混合”的极端 (即第一个和最后一个元素)。证明的关键在于对两端元素的区分 (与假设矛盾) 蕴含着对相邻元素的区分, 而由后者又能得到区分来自两个原始分布的单个样本的程序 (与两个分布由单个样本不可区分的假设矛盾)。具体细节如下。

由反证法, 假设 D 能区分来自于 X_n 的 $s(n)$ 个样本与来自 Y_n 的 $s(n)$ 个样本, 其区分缝隙为 $\delta(n)$ 。将第 i 个元素记作 H_n^i (即 $H_n^i = (X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(i)}, Y_n^{(i+1)}, Y_n^{(i+2)}, \dots, Y_n^{(s(n))})$), 则 D 能以缝隙 $\delta(n)$ 区分两端的元素 (即 H_n^0 和 $H_n^{s(n)}$)。那么, D 能区分任意一对相邻元素 (即对于随机选择的 i , D 能区分 H_n^i 与 H_n^{i+1}) 的缝隙至少为 $\delta(n)/s(n)$, 具体原因如下:

$$\begin{aligned} & E_{i \in \{0, 1, \dots, s(n)-1\}} \left[\Pr[D(H_n^i) = 1] - \Pr[D(H_n^{i+1}) = 1] \right] \\ &= \frac{1}{s(n)} \cdot \sum_{i=0}^{s(n)-1} \left(\Pr[D(H_n^i) = 1] - \Pr[D(H_n^{i+1}) = 1] \right) \\ &= \frac{1}{s(n)} \cdot \left(\Pr[D(H_n^0) = 1] - \Pr[D(H_n^{s(n)}) = 1] \right) = \frac{\delta(n)}{s(n)} \end{aligned} \quad (8.7)$$

论证中的关键步骤是将区分相邻元素的能力转化为区分原始空间单个样本的能力 (从而得出矛盾)。事实上, 利用 D 可得到单个样本的区分器 D' : 给定单个样本, 算法 D' 随机选择 $i \in \{0, 1, \dots, s(n)-1\}$, 生成第一个分布的 i 个样本, 以及第二个分布的 $s(n) - i - 1$ 个样本, 并将输入样本放入位置 $i+1$ 时, 对得到的对 $s(n)$ 个样本调用 D , 返回 D 的输出。也就是说, 对于输入 z 以及选择的索引 i , 算法 D' 对服从分布 $(X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(i)}, z, Y_n^{(i+2)}, Y_n^{(i+3)}, \dots, Y_n^{(s(n))})$ 的样本调用 D 。因此, D' 的构造依赖于两个概率空间都可在多项式时间内构造的假设。对 D' 的分析基于以下两个事实:

(1) 当输入服从分布 X_n , 且选择的索引 $i \in \{0, 1, \dots, s(n)-1\}$ 时, 算法 D' 的行为与 $D(H_n^{i+1})$

① 两个概率空间都可在多项式时间内构造, 这是一个基本要求, 见习题 8.10。

② 更多细节可见参考文献[91]中 3.2.3 节。

相近, 因为 $(X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(i)}, X_n, Y_n^{(i+2)}, Y_n^{(i+3)}, \dots, Y_n^{(s(n))}) \equiv H_n^{i+1}$ 。

(2) 当输入服从分布 Y_n , 索引 $i \in \{0, 1, \dots, s(n)-1\}$ 时, 算法 D' 的行为与 $D(H_n^i)$ 相近, 因为 $(X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(i)}, Y_n, Y_n^{(i+2)}, Y_n^{(i+3)}, \dots, Y_n^{(s(n))}) \equiv H_n^i$ 。

所以, D' 的区分缝隙 (区分 X_n 与 Y_n) 可由式 (8.7) 求出, 命题得证 (因为由反证法, 假设命题结论不成立时, 会得到与假设不一致的矛盾)。

混合技术——小资料

混合技术是一种特殊类型的“可约性方法”, 其中利用基本概率空间的计算不可区分性来证明复合概率空间的计算不可区分性。实际的归约从另一个方向实现: 将对基本概率空间的高效区分归约为对复合空间的高效区分, 归约中大量使用混合分布。以下关于混合构造的 3 个性质十分重要。

(1) 复合空间与两端元素相冲突。这是一条基本性质, 因为目标是证明复合概率空间的性质 (即不可区分性), 而归约本身针对着两端元素。在命题 8.6 的证明中, 两端元素 (即 H_n^0 和 $H_n^{s(n)}$) 与复合概率空间冲突, 后者表示两个基本概率空间之一的 $s(n)$ -长样本序列。

(2) 基本空间可以高效地映射到相邻元素。这个性质也很重要, 因为最初的假设与基本空间相关 (即其不可区分性), 而归约本身直接指向相邻元素。因此, 需要将关于基本空间的信息 (即计算不可区分性) 转化为关于任意一对相邻元素的信息 (即计算不可区分性)。典型地, 可将来自基本分布的串高效地转化为来自混合分布的串, 使得转化中将第一个基本分布映射为一个元素, 将第二个基本分布映射为相邻元素。

在命题 8.6 的证明中, 基本空间 (X_n 和 Y_n) 可以高效地转化为相邻元素 (即分别映射到 H_n^{i+1} 和 H_n^i)。在这种情况下, 转化的效率取决于两个基本空间都可在多项式时间内构造的假设。

(3) 元素的数量很少 (多项式个)。这是为了将两端元素的计算不可区分性归约为任意一对相邻混合的计算不可区分性。典型地, 证明中建立的“区分缝隙”丢掉了与元素数量成比例的一个因子。这是由于两端元素间的缝隙上限是相邻元素间缝隙之和。

在命题 8.6 证明中, 元素的数量为 $s(n)$, 而上述丢失的参数体现于式 (8.7) 中。

注意: 在混合方法中, 针对复合空间的区分算法可以用任意的混合空间分析甚至调用。读者也许会对算法“并非设计的工作于这些混合分布之上” (只工作于两端元素) 感到困扰。然而, 算法就是算法: 一旦构造出了算法, 就可以任意选择输入并调用算法, 并对任意输入分布分析算法的性能。

8.2.4 扩张函数的放大

注意: 伪随机数发生器的定义中 (即定义 8.1) 对扩张函数只有最低要求, 即只要求输出长度大于输入长度。人们需要的是有更大扩张的伪随机数发生器, 首先由于扩张决定了算法随机性需求的减少, 这体现于构造 8.2 中。具有任意扩张函数的伪随机数发生器 (特别是具有最小扩张 $l_1(k) \stackrel{\text{def}}{=} k+1$) 易转化为具有 (多项式界) 期望扩张 l 的伪随机数发生器 (见构造 8.7) (另一方面, 由于要求伪随机数发生器运行于多项式时间 (定义 8.1), 其扩张函数必须是多项式界的)。

构造 8.7 设 G_1 为具有扩张函数 $l_1(k) = k + 1$ 的伪随机数发生器, l 为任意多项式时间扩张函数, 令

$$G(s) \stackrel{\text{def}}{=} \sigma_1 \sigma_2 \cdots \sigma_{l(|s|)} \quad (8.8)$$

其中 $x_0 = s$ 而 $x_i \sigma_i = G_1(x_{i-1})$, $i = 1, 2, \dots, l(|s|)$ (即 σ_i 是 $G_1(x_{i-1})$ 的最后一个比特, 而 x_i 为 $G_1(x_{i-1})$ 的 $|s|$ -比特前缀)。

当然, G 可在多项式时间内计算, 且具有扩张 l 。习题 8.11 中给出了另一种构造。

命题 8.8 设 G_1 和 G 如构造 8.7, 则 G 构成一个伪随机数发生器。

证明概要: ①可用 8.2.3 节中的混合技术来证明。这里 (对 $i = 0, 1, \dots, l(k)$) 考虑由下式定义的混合分布 H_k^i (图 8.2), 即

$$H_k^i \stackrel{\text{def}}{=} U_i^{(1)} \cdot g_{l(k)-i}(U_k^{(2)})$$

其中 “ $\cdot$ ” 表示串的级联, $g_j(x)$ 表示 $G(x)$ 的 j 比特前缀, $U_i^{(1)}$ 和 $U_k^{(2)}$ 为独立均匀分布 (分别在 $\{0, 1\}^i$ 和 $\{0, 1\}^k$ 上)。两端元素 (即 H_k^0 和 H_k^k) 对应于 $G(U_k)$ 和 $U_{l(k)}$, 而区分相邻元素的能力可转化为区分 $G_1(U_k)$ 和 U_{k+1} 的能力。具体细节如下。

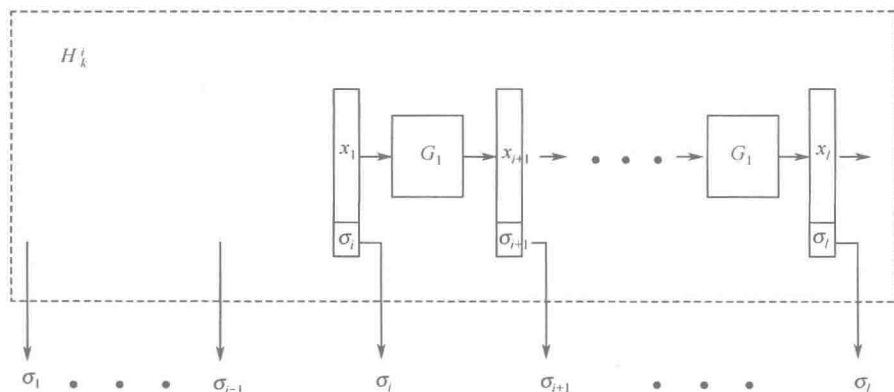


图 8.2 对扩张放大的分析——第 j 个元素

主要证明相邻元素是无法区分的。②用反证法, 假设算法 D 能区分 H_k^i 与 H_k^{i+1} 。首先更深入地观察这些元素。注意: 当 $j \geq 1$ 时, 有 $g_j(s) \equiv (\sigma, g_{j-1}(x))$, 其中 $x\sigma = G_1(s)$ 。将 x 的前 $|x| - 1$ 比特记作 $F(x)$, 最后一比特记作 $L(x)$, 可以写成 $g_j(s) \equiv (L(G_1(s)), g_{j-1}(F(G_1(s))))$ 及 $(U_i^{(1)}, U_k^{(2)}) \equiv (L(U_{k+1}), F(U_{k+1}))$, 则有

$$\begin{aligned} H_k^i &= U_i^{(1)} \cdot g_{l(k)-i}(U_k^{(2)}) \\ &\equiv (U_i^{(1)}, L(G_1(U_k^{(2)})), g_{(l(k)-i)-1}(F(G_1(U_k^{(2)})))) \end{aligned}$$

① 更多细节详见参考文献[91]中 3.3.3 节。

② 例照 (在使用混合技术时) 对两端元素的区分 (分别与 $G(U_k)$ 和 $U_{l(k)}$ 冲突) 蕴含着对相邻元素中一对随机元素的区分。因此, 下面的分析可应用于随机的 i (属于 $\{0, 1, \dots, k-1\}$), 而整个分析将利用类似于式 (8.7) 的表达式。

$$\begin{aligned} H_k^{i+1} &= U_{i+1}' \cdot g_{l(k)-i-1}(U_k^{(2)}) \\ &\equiv \left(U_i^{(1)}, L(U_{k+1}^{(2)}), g_{l(k)-i-1}(F(U_{k+1}^{(2)})) \right) \end{aligned}$$

现在, 结合 $U_i^{(1)}$ 的产生以及 $g_{l(k)-i-1}$ 的计算与区分器 D , 可以区分分布 $(F(G_1(U_k^{(2)})), L(G_1(U_k^{(2)}))) \equiv G_1(U_k)$ 与 $(F(U_{k+1}^{(2)}), L(U_{k+1}^{(2)})) \equiv U_{k+1}$, 这与 G_1 的伪随机性矛盾。

具体地, 对于输入 $x \in \{0,1\}^{k+1}$, 均匀选择 $r \in \{0,1\}^i$ 并输出 $D(r \cdot L(x) \cdot g_{l(k)-i-1}(F(x)))$ 。对区分器的分析基于以下两个事实:

(1) 当给定的输入服从分布 $G_1(U_k)$ 时, 对输入 $(U_i', L(G_1(U_k)), g_{l(k)-i-1}(F(G_1(U_k)))) \equiv H_k^i$ 调用算法 D 。

(2) 当给定的输入服从分布 U_{k+1} 时, 对输入 $(U_i', L(U_{k+1}), g_{l(k)-i-1}(F(U_{k+1}))) \equiv H_k^{i+1}$ 调用算法 D 。

因此, 对于输入 $G_1(U_k)$, 输出 1 的概率为 $\Pr[D(H_k^i)=1]$ (或 $\Pr[D(H_k^{i+1})=1]$)。所以, 对相邻元素的区分蕴含了对 $G_1(U_k)$ 和 U_{k+1} 的区分。□

结论

在上文中, 如果仅考虑伪随机数发生器的存在性, 则在定义 8.1 的意义下, 可以忽略特定的扩张函数。

8.2.5 构造

本节对构造方法进行综述, 这些构造均将单向函数形式的计算困难性“转化”为伪随机数发生器。回顾单向函数的定义, 称一个多项式时间计算的函数是单向的, 如果任意高效算法对其求逆的成功概率可以忽略 (见定义 7.1 及 7.1 节中的进一步讨论)。实际中使用的是这些函数的困难核心谓词, 读者可查阅 7.1.3 节的相关内容。不严格地, 一个多项式时间计算的谓词 b 称为函数 f 的困难核心, 如果给定 $f(x)$, 任意高效算法猜测 $b(x)$ 的成功概率约为 $1/2$ 。回顾第 7 章中提到的 (定理 7.7), 对任意单向函数 f , x 与 r 的内积模 2 是 $f'(x, r) = (f(x), r)$ 的困难核心。

命题 8.9 (伪随机数发生器的一种简单构造) 设多项式时间可计算的函数 f 为双射且保持长度, b 为 f 的困难核心谓词, 则 $G(s) \stackrel{\text{def}}{=} f(s) \cdot b(s)$ 是一个伪随机数发生器。

证明概要: ①考虑均匀选择的 $s \in \{0,1\}^n$, 首先要注意由于 f 诱导出集合 $\{0,1\}^n$ 上的一个置换, 所以 $G(s)$ 的 n 比特前缀在 $\{0,1\}^n$ 上均匀分布。因此, 命题的证明可归结为证明如果能区分 $f(s) \cdot b(s)$ 与 $f(s) \cdot \sigma$, 其中 σ 为一个随机比特, 则与 b 是 f 的困难核心 (即不能由 $f(s)$ 预测 $b(s)$) 的假设矛盾。直观原因是这个假想的区分器也能区分 $f(s) \cdot b(s)$ 与 $f(s) \cdot \overline{b(s)}$, 其中 $\overline{\sigma} = 1 - \sigma$, 而从对 $f(s) \cdot b(s)$ 与 $f(s) \cdot \overline{b(s)}$ 的区分可得到一个算法来由 $f(s)$ 预测 $b(s)$ 。具体细节如下。

① 更多细节详见参考文献[91]中 3.3.4 节。

我们从可能的区分器 D 开始, 并令

$$\delta(k) \stackrel{\text{def}}{=} \Pr[D(G(U_k))=1] - \Pr[D(U_{k+1})=1]$$

不失一般性, 假设 $\delta(k)$ 非负 (对无穷多个 k)。观察到 $G(U_k) = f(U_k) \cdot b(U_k)$, 且 U_{k+1} 的分布与这样一个随机变量相同, 该变量取值为 $f(U_k)b(U_k)$ 的概率是 $1/2$, 取值为 $f(U_k)\overline{b(U_k)}$ 的概率也是 $1/2$, 从而有

$$\Pr[D(f(U_k)b(U_k))=1] - \Pr[D(f(U_k)\overline{b(U_k)})=1] = 2\delta(k)$$

关键在于 D 可以有效地区分 (以缝隙 $2\delta(k)$) 最后一个比特为 $b(U_k)$ 或为 $\overline{b(U_k)}$, 这种区分能力可以转化为利用 $f(U_k)$ 来预测 $b(U_k)$ 的值。事实上, 考虑这样一个算法 A , 对于输入 y , 均匀选择 $\sigma \in \{0,1\}$, 调用 $D(y\sigma)$, 并当 $D(y\sigma)=1$ 时输出 σ , 否则输出 $\overline{\sigma}$, 从而

$$\begin{aligned} & \Pr[A(f(U_k))=b(U_k)] \\ &= \Pr[D(f(U_k) \cdot \sigma)=1 \wedge \sigma=b(U_k)] + \Pr[D(f(U_k) \cdot \sigma)=0 \wedge \sigma=\overline{b(U_k)}] \\ &= \frac{1}{2} \cdot \left(\Pr[D(f(U_k) \cdot b(U_k))=1] + \left(1 - \Pr[D(f(U_k) \cdot \overline{b(U_k)})=1] \right) \right) \end{aligned}$$

这个概率等于 $(1+2\delta(k))/2$ 。这与 b 是 f 的困难核心的假设矛盾, 命题得证。 $\square$

结合定理 7.7, 命题 8.9 与构造 8.7, 可以得到如下结论。

定理 8.10 (伪随机数发生器存在的充分条件) 如果存在保持长度且为双射的单向函数, 则对任意多项式界的扩张函数 l , 存在具有扩张 l 的伪随机数发生器。

思考

命题 8.9 证明的关键部分是证明 $G(U_k)$ (下一比特) 的不可预测性蕴含了其伪随机性。以下给出定理 8.10 的另一种证明, 其中明确地证明了: (下一比特的) 不可预测性等价于伪随机性。

8.2.5.1 另一种表示

进一步观察将构造 8.7 和命题 8.9 相结合得到的伪随机数发生器。对于扩张函数 $l: \mathbb{N} \rightarrow \mathbb{N}$, 单向且为双射的函数 f , b 为其困难核心, 有

$$G(s) \stackrel{\text{def}}{=} \sigma_1 \sigma_2 \cdots \sigma_{l(|s|)} \quad (8.9)$$

其中 $x_0 = s$ 且 $x_i \sigma_i = f(x_{i-1})b(x_{i-1})$, $i=1,2,\dots,l(|s|)$ 。将 f 对于输入 x 迭代 i 次之后的结果记作 $f^i(x)$ (即 $f^i(x) = f^{i-1}(f(x))$, 且 $f^0(x) = x$), 将式 (8.9) 重新写成如下形式:

$$G(s) \stackrel{\text{def}}{=} b(s) \cdot b(f(s)) \cdots b(f^{l(|s|)-1}(s)) \quad (8.10)$$

利用 (下一比特的) 不可预测性, 可分两步证明 G 的伪随机性。空间 $\{Z_k\}_{k \in \mathbb{N}}$ 称为不可预测的, 如果对任意概率多项式时间机器, 给定 Z_k 的 (随机) 前缀^①, 无法以明显大于 $1/2$ 的概率来预测 Z_k 的下一比特。具体而言, 要证明以下两个结论。

(1) **一般性结论。** 一个空间是伪随机的当且仅当其是不可预测的。回顾前面将概率空间

^① 为简单起见, 用具有随机长度的前缀 (在 $\{0,1,\dots,|Z_k|-1\}$ 上均匀分布) 定义不可预测性。在更通用的定义中, 允许预测者确定其可读前缀的长度。这种似乎更强的定义实际上等价于我们使用的定义, 因为两者都与伪随机性等价。

的伪随机性定义为与均匀分布的计算不可区分性（在足够长的比特串上）。

显然，伪随机性蕴含了多项式时间不可预测性，但在这里实际上要证明另一个方向，而这并不显然。然而利用混合方法，可以证明（下一比特的）不可预测性蕴含了与均匀空间的不可区分性。细节可见习题 8.12。

(2) 特殊的结论。空间 $\{G(U_k)\}_{k \in \mathbb{N}}$ 是从右向左不可预测的。等价地， $G'(U_n)$ 是多项式时间不可预测的（从左向右（习惯如此）），其中 $G'(s) = b(f^{l(s)-1}(s)) \cdots b(f(s)) \cdot b(s)$ 是 $G(s)$ 的翻转。

利用 f 诱导出 $\{0,1\}^n$ 上的置换这一事实，并观察到 $G'(U_k)$ 的 $j+1$ -比特前缀与 $b(f^j(U_k)) \cdots b(f(U_k)) \cdot b(U_k)$ 分布相同。因此，由基于 $G'(U_n)$ 的 j -比特前缀预测第 $j+1$ 比特的算法出发，可得到一个算法由 $f(U_k)$ 猜测 $b(U_n)$ 。细节见习题 8.14。

显然， G 为伪随机数发生器当且仅当 G' 是一个伪随机数发生器（习题 8.13）。式 (8.10) 常被称为 Blum-Micali 构造。^①

8.2.5.2 伪随机数发生器存在性的一般条件

在上述讨论中，给定一个单向、双射且保持长度的函数，易构造一个伪随机数发生器。实际上，要求函数是双射（且保持长度）并无必要，但是目前已知的（通用）构造都相当复杂。

定理 8.11（伪随机数发生器的存在性） 伪随机数发生器存在当且仅当单向函数存在。

为证明伪随机数发生器的存在性蕴含了单向函数的存在性，考虑具有扩张函数 $l(k) = 2k$ 的伪随机数发生器 G 。对于 $x, y \in \{0,1\}^k$ ，定义 $f(x, y) \stackrel{\text{def}}{=} G(x)$ ，则 f 可在多项式时间内计算（且保持长度）。 F 必为单向的，否则可以对 f 求逆，并由结果区分 $G(U_k)$ 与 U_{2k} 。在分布 $f(U_{2k})$ 上对 f 求逆对应于 $G(U_k)$ 上的操作，而 U_{2k} 中存在逆元的概率可忽略。

证明定理 8.11 的另一个方向需要基于任意的单向函数构造伪随机数发生器。由于已知的证明方法都相当复杂，我们只粗略简单介绍相关的证明思想。这些证明中大量使用了 Hash 函数（两两独立的 Hash 函数，见附录 D.2）。

首先，要注意，（当 f 非双射时）概率空间 $f(U_k)$ 可能不是伪随机的，从而构造 8.9（即 $G(s) = f(s)b(s)$ ，其中 b 为 f 的困难核心）并不能直接应用。已知的构造思想是利用 Hash 函数将 $f(U_k)$ 映射为几乎均匀的串，长度与其熵(entropy)相关。^②但是“将 $f(U_k)$ 通过 Hash 映射为长度相当于其熵的串”意味着长度会缩小，即 $k' < k$ ，这样就无法对 k 比特随机种子进行扩张。因此，第二种构造思想是从 U_k 中提取丢失的 $k - k'$ 比特。方法是求 U_k 的 Hash 值，然而 $k - k'$ 比特的 Hash 值并不会使求逆的工作更容易。这些思想实现起来极为复杂，所以人们更倾向于采用其他构造。

8.2.6 非一致的强伪随机数发生器

前面讲过，用伪随机序列代替真随机序列不会影响使用这些序列的高效计算。命题 8.3

^① 鉴于该方法已被广泛应用，我们在正文中并未给出其出处。这一构造来自参考文献[41]。

^② 在确保 $f(U_k)$ 取某个值的概率的对数接近于 $f(U_k)$ 的熵时，可以实现这一点。具体而言，给定任意一个单向函数 f' ，首先利用 f' 的足够多副本的“直积”来构造 f 。例如，对于 $x_1, x_2, \dots, x_{k/2} \in \{0,1\}^{k/2}$ ，令 $f(x_1, x_2, \dots, x_{k/2}) \stackrel{\text{def}}{=} f'(x_1), f'(x_2), \dots, f'(x_{k/2})$ 。

给出了这个断言的形式化描述, 其中涉及到随机算法, 并在计算中使用“基本输入”和第二个“随机输入”。命题 8.3 认为不可能找到一个基本输入, 使得将真随机的第二输入用伪随机数代替时, 能明显影响算法的最终输出。然而, 这并不意味着这些基本输入不存在(只是难找到而已)。因此, 命题 8.3 不能提供(在最坏情况下)对复杂性类如 BPP 类的“去随机化”。^① 为得到这种结论, 需要定义更强的伪随机数发生器。也就是说, 要求伪随机数发生器可以欺骗任意多项式规模电路(见 1.2.4.1 节), 而不仅仅是所有概率多项式时间算法。^②

定义 8.12 (强伪随机数发生器——欺骗电路) 称确定的多项式时间算法 G 为非一致的强伪随机数发生器, 如果存在扩张函数, $l: \mathbf{N} \rightarrow \mathbf{N}$, 使得对任意多项式规模电路簇 $\{C_k\}_{k \in \mathbf{N}}$, 任意正多项式 p 及足够大的 k , 有

$$\left| \Pr[C_k(G(U_k))=1] - \Pr[C_k(U_{l(k)})=1] \right| < \frac{1}{p(k)}$$

从接受建议的多项式时间机器(见 3.1.2 节)出发, 可以得到另一种定义。利用这种伪随机数发生器, 可以对 BPP 类“去随机化”。

定理 8.13 (BPP 类的去随机化) 如果非一致的强伪随机数发生器存在, 则 BPP 包含于 $\bigcap_{\varepsilon > 0} D_{\text{TIME}}(t_\varepsilon)$ 中, 其中 $t_\varepsilon(n) \stackrel{\text{def}}{=} 2^{n^\varepsilon}$ 。

证明概要: 对任意 $S \in BPP$ 及任意 $\varepsilon > 0$, 令 A 表示 S 的判定程序, G 表示非一致的强伪随机数发生器, 可将 n^ε 比特随机种子扩张为 $\text{poly}(n)$ -长的序列(作为算法 A 的第二输入, 其基本输入长度为 n)。结合 A 与 G , 可得到算法 $A' = A_G$ (如构造 8.2)。我们断言 A 与 A' 只对至多有限个输入的判定(在期望概率上)明显不同, 否则, 可利用这些输入(及 A)构造(非一致的)多项式规模电路簇, 能区分 $G(U_{n^\varepsilon})$ 与 $U_{\text{poly}(n)}$, 这与关于 G 的假设矛盾。具体而言, 如果 A 与 A' 对于输入 x 的判定不同, 则通过令 $C_x(r) = A(x, r)$ ^③, 可得到电路 C_x , 能区分 $G(U_{|x|^\varepsilon})$ 与 $U_{\text{poly}(|x|)}$ 。合并这些“坏”的输入, 可得到判定 S 的概率多项式算法, 随机复杂度为 n^ε 。

最后, 对 2^{n^ε} 个可能的随机序列(G 的种子)中的每一个模仿 A' , 并由大数表决, 可得到一个符合要求的确定算法 A'' 。即令 $A'(x, r)$ 表示 A' 对输入 x 和随机数 $r \in \{0, 1\}^{n^\varepsilon}$ 的输出, 则 A'' 对所有 $r \in \{0, 1\}^{n^\varepsilon}$ 调用 $A'(x, r)$, 并当且仅当这 2^{n^ε} 个调用中大多数返回 1 时, A'' 输出 1。□

对 BPP 类去随机化更强的结论在 8.3 节给出。

构造非一致的强伪随机数发生器

非一致的强伪随机数发生器(定义 8.12)可以利用单向函数构造, 这些函数由任意非一

① 事实上, 命题 8.3 给出了对 BPP 平均情况下的去随机化。特别地, 对任意多项式时间可构造的空间 $\{X_n\}_{n \in \mathbf{N}}$, 任意布尔函数 $f \in BPP$ 及 $\varepsilon > 0$, 存在一个随机复杂度为 $r_\varepsilon(n) = n^\varepsilon$ 的随机算法 A' , 使得 $A'(X_n) \neq f(X_n)$ 的概率可以忽略。利用定理 8.13 证明中的方法, 可以得到相应的 ($\exp(r_\varepsilon)$ -时间)确定算法 A'' , 使得 $A''(X_n) \neq f(X_n)$ 的概率也可忽略, 这里概率的取值范围是基本输入(即 X_n)的分布。相反, 最坏情况下的去随机化, 表示为断言 $BPP \subseteq D_{\text{TIME}}(2^{r_\varepsilon})$, 要求 $A''(X_n) \neq f(X_n)$ 的概率为 0。

② 当然, 定义 8.12 中的强伪随机数发生器符合伪随机数发生器的基本定义(定义 8.1), 见习题 8.15。这里计算不可(由电路)区分的概念比定义 8.4 中更强, 其在多个样本下是不变的(无论概率空间如何构造), 细节见习题 8.16。

③ 事实上, 由命题 8.3 的证明, 搜索器 F 包含一个非一致的多项式规模电路簇, 可输出嵌入其中的“有问题”的基本输入, 因而相应的区分器是非一致的。

致的多项式规模电路难求逆（定义 7.3），而非利用概率多项式时间机。事实上，此时的构造比一致情形（即定理 8.11 证明中采用的构造）要简单。

8.2.7 更强的概念及思考

本小节首先给出伪随机数发生器两种更强的定义，并在结束时讨论各种概念问题。

8.2.7.1 更强的（一致复杂性）概念

以下两个概念是对伪随机数发生器标准定义（定义 8.1）的强化。这些变形（定义 8.12 的强化）的非一致性版本也值得关注。

欺骗更强的区分器

对定义 8.1 的一种强化是明确量化区分器使用的资源（及成功缝隙）。我们选择将这些量定义为种子长度（即 k ）的函数，而非待检查串长度（即 $l(k)$ ）的函数。对于一类时间界限 $\mathcal{T}$ （如 $\mathcal{T} = \left\{ t(k) \stackrel{\text{def}}{=} 2^{c\sqrt{k}} \right\}_{c \in \mathbb{N}}$ ），以及一类不可忽略的函数 $\mathcal{F}$ （如 $\mathcal{F} = \left\{ f(k) \stackrel{\text{def}}{=} 1/t(k) : t \in \mathcal{T} \right\}$ ），

称伪随机数发生器 G 是 $(\mathcal{T}, \mathcal{F})$ -强化的，如果对任意运行时间为 $\mathcal{T}$ 的函数（应用于 k ）的概率算法 D ， $\mathcal{F}$ 中任意函数 f ，以及足够大的 k ，有

$$\left| \Pr[D(G(U_k))=1] - \Pr[D(U_{l(k)})=1] \right| < f(k)$$

类似的强化可用于单向函数的定义。这样做暴露了定理 8.11 证明中构造方法的一个弱点：它只暗示对某些 $\varepsilon > 0$ （可令 $\varepsilon = 1/8$ ），任意的 $\mathcal{T}$ 和 $\mathcal{F}$ ，“ $(\mathcal{T}, \mathcal{F})$ -强化单向函数”的存在性蕴含了 $(\mathcal{T}', \mathcal{F}')$ -强化伪随机数发生器的存在性，其中 $\mathcal{T}' = \left\{ t'(k) \stackrel{\text{def}}{=} t(k^\varepsilon) / \text{poly}(k) : t \in \mathcal{T} \right\}$ ，而

$\mathcal{F}' = \left\{ f'(k) \stackrel{\text{def}}{=} \text{poly}(k) \cdot f(k^\varepsilon) : f \in \mathcal{F} \right\}$ 。我们希望对于 $\mathcal{T}' = \left\{ t'(k) \stackrel{\text{def}}{=} t(\Omega(k)) / \text{poly}(k) : t \in \mathcal{T} \right\}$

及 $\mathcal{F}' = \left\{ f'(k) \stackrel{\text{def}}{=} \text{poly}(k) \cdot f(\Omega(k)) : f \in \mathcal{F} \right\}$ 有类似的结论。

伪随机函数

伪随机函数发生器可由短的随机数种子高效生成伪随机序列。伪随机函数（在附录 C.3.3 中定义）的功能则更强大：能高效地直接访问一个庞大的、甚至无法逐比特扫描的伪随机序列。即基于（随机的） k 比特种子，可以直接访问长为 2^k 的序列。换句话说，伪随机函数是确定的多项式时间算法，将 k 比特长的种子 s 和 k 比特变量 x 映射为一个值 $f_s(x)$ ，使得对于均匀分布的 $s \in \{0,1\}^k$ ，函数 f_s 对任意 $\text{poly}(k)$ -时间的观察者而言都是随机的，该观察者可以任选变量值并询问 f_s 。因此，伪随机函数可在任意高效的应用中（如最著名的，在密码学中）代替真正的随机函数。伪随机函数可由任意伪随机数发生器构造（见定理 C.8），它们在密码学中得到了大量应用（见附录 C.3.3、附录 C.5.2 和附录 C.6.2）。伪随机函数也可用于计算学习理论^[232]中，推导一些负面结论，还可用于电路复杂性的研究中（见参考文献[189]）。

8.2.7.2 概念上的思考

对于上述随机性研究中的计算方法，在概念上要强调几个方面，它们与一般模式在一些实例（8.3 节中的除外）中是相同的。

行为主义与存在主义

用计算方法研究随机性本质上具有行为主义的特征，为理解这一点，可将该方法与 Kolmogorov-Chaitin 方法进行比较。不严格地，称一个串是柯尔莫果洛夫随机的 (Kolmogorov-random)，如果其长度等于生成该串的最短程序的长度。这个最短程序可以看作是对该串描述现象的“真正解释”。因此，一个柯尔莫果洛夫随机的串不存在比其自身更简单 (更短) 的解释。考虑一种现象的最简单解释属于存在主义方法。相反，考虑现象在某种设备 (或观测) 中的效果，如伪随机性定义中所体现的，则属行为主义方法。进一步地，存在一些非均匀的概率分布 (且并不与均匀分布统计接近)，仍与均匀分布不可区分 (由任意高效设备)。因此，从存在主义观点来看差别较大的分布，从行为主义观点来看却是等价的 (在计算不可区分的定义下)。

随机性的相对观点

我们已经用观察者的术语定义了伪随机性。具体地，考虑高效 (即多项式时间) 的观察者集合，将该集合中任意观察者认为是随机的对象定义为具有伪随机性。后面将对这些观察者进行限定 (如空间受限的多项式时间观察者，甚至更严格的受限情况，如限定观察者只能使用特定类型测试，如线性测试或命中测试)。从每一类受限的观察者会得到伪随机性的不同概念。进一步地，(伪随机性的) 一般模式明确地针对着非均匀的分布，而从某些观察者角度看也确实如此。因此，对伪随机性的总体研究方法是相对的和主观的 (取决于观察者能力)。

随机性与计算困难性

伪随机性与计算困难性担负双重角色：伪随机性的一般模式依赖于一个事实，即对观察者在计算上进行限制后，能得到一些非均匀但与均匀分布不可区分的分布。因此，整个方法的重点就是区分伪随机分布与真随机分布在计算上的困难性。进一步地，伪随机数发生器的许多构造或者依赖于假设，或者依赖于计算困难性 (即某些计算在特定的类中是困难的)。例如，用单向函数构造通用的伪随机数发生器 (运行于多项式时间且能欺骗所有多项式时间观察者)。类似地，将在 8.3.3.1 节中看到，奇偶校验函数对于多项式规模、常数深度的电路来说是困难的，这一事实可用来生成 (极不均匀地) 能欺骗这些电路的序列。

随机性与预测性

伪随机性与不可预测性 (由高效程序) 之间的联系对于分析一些构造至关重要 (见 8.2.5 节和 8.3.2 节)。我们希望读者重视这种联系。

8.3 对时间复杂性类的去随机化

回顾定理 8.13 证明中的去随机化过程。首先，用一个伪随机数发生器来降低 BPP-算法的随机性复杂度，再通过扫描该发生器所有可能的种子来去随机化。这个过程的关键在于，要求伪随机数发生器运行于种子长度的多项式时间是无意义的。相反，只要求发生器运行于种子长度的指数时间就足够了。进一步地，此时发生器的运行时间可能超过了算法的运行时间，这意味着发生器只需要骗过运行步骤少于该发生器的区分器即可。这引出如下的正则去随机性发生器 (Canonical Derandomizer)。

8.3.1 正则去随机性发生器

为了对概率多项式时间算法 A “去随机化”，首先要得到一个功能上等价的算法 A_G (见

构造 8.2), 其随机性复杂度远远小于 A 。对于 A 的所有输入 (除有限个以外), 算法 A_G 必须保持 A 所有的输入-输出对应关系。因此, 相关的区分器集 (在定理 8.13 证明中所提到的) 通过将所有可能输入嵌入到 A 中得到的所有可能电路集。其中一个电路, 记作 C_x , 模仿算法 A 对于输入 x 的运行过程, 并将电路的输入作为算法的内部随机数 (即 $A(x, r) = C_x(r)$)。进一步地, 电路 C_x 的规模是 A 对输入 x 运行时间的平方, 而 C_x 的输入长度等于 A (对输入 x) 的运行时间。^①因此, 电路 C_x 的规模是其自身输入的平方, 而使用的伪随机数发生器 (即 G) 要能欺骗每一个这种电路。前面提到, 可以允许发生器运行于指数时间 (即时间复杂度是其自身输入 (即种子) 长度的指数), ^②从而得到以下定义。

定义 8.14 (对 $\text{BP}_{\text{TIME}}(\cdot)$ 去随机化的伪随机数发生器) ^③ 设 $l: \mathbf{N} \rightarrow \mathbf{N}$ 为单调递增函数, 具有扩张 l 的正则去随机发生器是一个确定算法 G , 满足以下两个条件。

(1) 对于输入的 k 比特种子, G 最多运行 $\text{poly}(2^k \cdot l(k))$ 步, 并输出长度为 $l(k)$ 的串。

(2) 对任意规模为 $l(k)^2$ 的电路 D_k , 有

$$\left| \Pr[D_k(G(U_k)) = 1] - \Pr[D_k(U_{l(k)}) = 1] \right| < \frac{1}{6} \quad (8.11)$$

电路 D_k 表示可能的区分器, 其输入 (从 $G(U_k)$ 或 $U_{l(k)}$ 中采样) 长度为 $l(k)$ 。对时间复杂度为 t 的算法 A 去随机化时, 上述的 $l(k)$ -比特串表示对长度为 $n = t^{-1}(l(k))$ 的 (原始) 输入调用 A 时, 一种可能的随机输出。因此, 对任意 $x \in \{0, 1\}^n$, 考虑电路 $D_k(r) = A(x, r)$, 其中 $|r| = t(n) = l(k)$, 注意到式 (8.11) 表明 $A_G(x) = A(x, G(U_k))$ 保持了 $A(x) = A(x, U_{l(k)})$ 的多数表决。另一方面, G 的时间复杂度暗示了对 $A_G(x)$ 直接而确定的实现需要时间为 $2^k \cdot (\text{poly}(2^k \cdot l(k)) + t(n))$, 其上限为 $\text{poly}(2^k \cdot l(k)) = \text{poly}(2^{t^{-1}(t(n))} \cdot t(n))$ 。这就得到了如下的 (有条件的) 去随机化结论。

命题 8.15 设 $l, t: \mathbf{N} \rightarrow \mathbf{N}$ 为单调递增函数, $l^{-1}(t(n))$ 表示使 $l(k) \geq t(n)$ 成立的最小整数 k 。如果存在一个扩张为 l 的正则去随机发生器, 则对任意可定时构造的 $t: \mathbf{N} \rightarrow \mathbf{N}$, 有 $\text{BP}_{\text{TIME}}(t) \subseteq D_{\text{TIME}}(T)$, 其中 $T(n) = \text{poly}(2^{l^{-1}(t(n))} \cdot t(n))$ 。

① 实际上, 我们假设算法 A 可以表示为一个图灵机, 而图灵机的标准实现是电路 (如定理 2.21 的证明中所体现的)。因此, 上述的电路 C_x 规模最多为 A 对输入 x 运行时间的平方, 这又表示 C_x 的规模至多是其自身输入长度的平方 (事实上, 通过构造更好的实现, 可使电路规模为 A 运行时间的线性函数^[180])。注意: 许多资料中人为地形成一种习惯, 令电路规模等于其输入长度, 如果对输入恰当填充, 可以证明这种习惯的合理性。

② 实际上, 定义 8.14 允许发生器运行于时间 $\text{poly}(2^k l(k))$ 而非 $\text{poly}(2^k)$ 。这是为了不把具有超指数扩张 (即 $l(k) = 2^{\omega(k)}$) 的发生器排除在外。然而 (见习题 8.18), 式 (8.11) 并不允许超指数的扩张 (甚至是不允许 $l(k) = \omega(2^k)$)。因此, 可以认为两个公式是等价的 (因为当 $l(k) = 2^{\omega(k)}$ 时, $\text{poly}(2^k l(k)) = \text{poly}(2^k)$)。

③ 固定一种计算模型, 用 $\text{BP}_{\text{TIME}}(t)$ 表示可由时间复杂度为 t 且具有双边错误 $1/3$ 的随机算法求解的判定问题类。令“门限区分缝隙”为 $1/6$ (式 (8.11) 中) 保证了如果 $\Pr[D_k(U_{l(k)}) = 1] \geq 2/3$ (或 $\Pr[D_k(U_{l(k)}) = 1] \leq 1/3$), 则 $\Pr[D_k(G(U_k)) = 1] > 1/2$ (或 $\Pr[D_k(G(U_k)) = 1] < 1/2$)。我们将看到, 这足以证明可在时间 T 内对 $\text{BP}_{\text{TIME}}(t)$ 去随机化, 其中 $T(n) = \text{poly}(2^{l^{-1}(t(n))} \cdot t(n))$ (且可以使用长为 $k = l^{-1}(t(n))$ 的种子)。

证明概要：只需仿照定理 8.13 的证明，其中又用到构造 8.2（给定任意随机算法 A 及发生器 G ，构造 8.2 能给出随机复杂度为 $l^{-1} \circ t$ 且时间复杂度为 $\text{poly}(2^{l^{-1} \circ t}) + t$ 的算法 A_G ）。^①观察到最终得到的确定算法的复杂度取决于 2^k 次调用 $A_G(x, s) = A(x, G(s))$ ，其中 $s \in \{0, 1\}^k$ ，且每次调用所需时间为 $\text{poly}(2^{l^{-1}(t(|x|))}) + t(|x|)$ 。因此，对于输入的 n 比特串，该确定程序运行于时间 $\text{poly}(2^{l^{-1}(t(n))} \cdot t(n))$ 之内。其正确性（在 2^k 次对 A_G 的调用中进行大数表决）可结合式 (8.11) 及假设 $\Pr[A(x)=1]$ 远离 $1/2$ 来证明。具体地，利用假设 $|\Pr[A(x)=1] - (1/2)| \geq 1/6$ ，对 $(A_G(x, s))_{s \in \{0, 1\}^k}$ 大数表决的结果为 1（等价地 $\Pr[A(x, G(U_k))=1] > 1/2$ ）当且仅当 $\Pr[A(x)=1] > 1/2$ （等价地， $\Pr[A(x, U_{l(k)})=1] > 1/2$ ）。事实上，这是由于式 (8.11)，并将其应用于电路 $C_x(r) = A(x, r)$ （规模最多为 $|r|^2$ ）。□

目标：

由命题 8.15，我们需要的是扩张尽可能大的正则去随机发生器。但扩张不能是超指数的（即必须有 $l(k) = O(2^k)$ ），因为存在规模为 $O(2^k \cdot l(k))$ 的电路使式 (8.11) 不成立（见习题 8.18），而当 $l(k) = \omega(2^k)$ 时，有 $O(2^k \cdot l(k)) < l(k)^2$ 。因此，这里的目标是构造具有扩张 $l(k) = 2^{\Omega(k)}$ 的正则去随机发生器，可以“对 BPP 类完全地去随机化”。

定理 8.16 如果存在一个扩张为 $l(k) = 2^{\Omega(k)}$ 的正则去随机发生器，则 $BPP = P$ 。

证明：利用命题 8.15，有 $BP_{\text{TIME}}(t) \subseteq D_{\text{TIME}}(T)$ ，其中 $T(n) = \text{poly}(2^{l^{-1}(t(n))} \cdot t(n)) = \text{poly}(t(n))$ 。■

思考

回顾正则去随机发生器 G 的定义中允许其时间复杂度 t_G 大于要欺骗的电路规模（即允许 $t_G(k) > l(k)^2$ ）。进一步地，还允许 $t_G(k) > 2^k$ 。因此，如果事实上 $t_G(k) = 2^{\Omega(k)}$ （如 8.3.2 节中的情况），则可在时间 $2^k \cdot t_G(k) = \text{poly}(t_G(k))$ 内通过尝试所有可能的种子来区分 $G(U_k)$ 和 $U_{l(k)}$ 。^②我们强调这里的区分器是一个一致性算法（其工作方式是对所有可能的种子调用 G ）。相反，对于通用的伪随机数发生器 G （如 8.2 节所讨论的），有 $t_G(k) = \text{poly}(k)$ ，而对任意多项式 p ，在时间 $p(t_G(k))$ 内无法区分 $G(U_k)$ 和 $U_{l(k)}$ 。

① 实际上，给定任意一个随机算法 A 和发生器 G ，利用构造 8.2 能得到一个算法 A_G ，定义为 $A_G(x, s) = A(x, G'(s))$ ，其中 $|s| = l^{-1}(t(|x|))$ ，而 $G'(s)$ 表示 $G(s)$ 的 $t(|x|)$ -比特前缀。为简单起见，这里假设 $l(|s|) = t(|x|)$ ，并用 G 来代替 G' 。注意到，给定 n ，可对 $i = 1, 2, \dots, k$ 调用 $G(i^i)$ ，求出 $k = l^{-1}(t(n))$ （利用 $l: \mathbb{N} \rightarrow \mathbb{N}$ 是单调递增的这一事实）。还要注意 $l(k) = O(2^k)$ 必须成立，从而可用 $\text{poly}(2^k)$ 来代替 $\text{poly}(2^k \cdot l(k))$ 。

② 注意：这个区分器与 G 为正则去随机发生器的假设并不矛盾，因为 $t_G(k) > l(k)$ 一定成立，而 $l(k) \leq 2^k$ 通常也成立（从而 $2^k \cdot t_G(k) > l(k)^2$ ）。

8.3.2 正则去随机性发生器的构造

正则去随机发生器可能比相应的区分器更复杂,这使得 8.2 节中的某些技术无法应用于本节。例如,扩张函数不能像 8.2.4 节中那样放大(见习题 8.17)。另一方面,下文提出的技术也不能用于 8.2 节。例如,对于某些正则去随机发生器(即构造 8.17 中的发生器),即使将种子给了区分器,发生器也仍是伪随机的。这种令人惊奇的现象实际上是由于区分器的时间复杂度限制了其无法对给定种子运行发生器。

8.3.2.1 构造及结论

与 8.2.5 节相似,以下给出的构造方法也将计算困难性转化为伪随机性,但这里的计算困难性和伪随机性与 8.2.5 节中形式不同。这里使用了可在指数时间内计算的布尔谓词,但对于某个指数函数 T ,该谓词是 T -不可近似的(见定义 7.9,以下将重述该定义)。换言之,假设存在布尔谓词和常数 $c, \varepsilon > 0$,使得对于除有限多个 m 之外,其余的谓词 $f: \{0,1\}^m \rightarrow \{0,1\}$ 可在时间 2^{cm} 内计算,但对任意规模为 $2^{\varepsilon m}$ 的电路 C ,有 $\Pr[C(U_m) = f(U_m)] < \frac{1}{2} + 2^{-\varepsilon m}$ (显然,在这里 $\varepsilon < c$)。回顾在假设 ε 的电路复杂度(几乎处处)为指数时,这样的谓词存在(见定理 7.19)。有了这些预备知识,下面构造具有指数扩张的正则去随机发生器。

构造 8.17 (Nisan-Wigderson 构造)^① 设 $f: \{0,1\}^m \rightarrow \{0,1\}$, $S_1, S_2, \dots, S_l$ 为 $\{1, 2, \dots, k\}$ 的 m -元素子集序列,则对 $s \in \{0,1\}^k$, 令

$$G(s) \stackrel{\text{def}}{=} f(s_{S_1}) f(s_{S_2}) \cdots f(s_{S_l}) \quad (8.12)$$

其中 s_S 表示 s 在 $S \subseteq \{1, 2, \dots, k\}$ 中比特位置上的投影,即对 $s = \sigma_1 \sigma_2 \cdots \sigma_k$ 和 $S = \{i_1, i_2, \dots, i_m\}$, 有 $s_S = \sigma_{i_1} \sigma_{i_2} \cdots \sigma_{i_m}$ 。

令 k 为变量, $l, m: \mathbb{N} \rightarrow \mathbb{N}$ 为 k 的函数,我们希望 G 为正则去随机发生器,且 $l(k) = 2^{\Omega(k)}$ 。一个(显然的)必要条件是这些子集必须不同,从而有 $m(k) = \Omega(k)$,故 f 一定可在指数时间计算。进一步地,集合序列 $S_1, S_2, \dots, S_{l(k)}$ 必可在时间 $\text{poly}(2^k)$ 内构造。直观上函数 f 应该是强不可近似的(即对某个指数函数 T ,不能 T -近似),而使用一组两两之间交集较小的集合也是可取的(因为这限制了 f 应用于不同输入时的交叠)。有趣的是,这些条件是充分条件。

定理 8.18 (对构造 8.17 的分析) 令 $\alpha, \beta, \gamma, \varepsilon > 0$ 为满足 $\varepsilon > (2\alpha/\beta) + \gamma$ 的常数,考虑函数 $l, m, T: \mathbb{N} \rightarrow \mathbb{N}$ 满足 $l(k) = 2^{\alpha k}$, $m(k) = \beta k$, 而 $T(n) = 2^{\varepsilon n}$, 假设以下两个条件成立。

(1) 存在一个指数时间计算的函数 $f: \{0,1\}^* \rightarrow \{0,1\}$ 是 T -不可近似的(见定义 7.9)。

(2) 存在一个指数时间计算的函数 $S: \mathbb{N} \times \mathbb{N} \rightarrow 2^{\mathbb{N}}$ 使得

① 对任意 k 和 $i \in [l(k)]$, 有 $S(k, i) \subseteq [k]$ 和 $|S(k, i)| = m(k)$;

② 对任意 k 和 $i \neq j$, 有 $|S(k, i) \cap S(k, j)| \leq \gamma \cdot m(k)$,

则利用构造 8.17 中定义的 G , 以及 $S_i = S(k, i)$, 可得到一个扩张为 l 的正则去随机发生器。

在证明定理 8.18 之前,要注意,对任意 $\gamma > 0$, 某些 $m(k) = \Omega(k)$ 和 $l(k) = 2^{\Omega(k)}$, 满足条件 (2) 的函数 S 确实存在,见习题 8.17。将这样一个函数 S 与定理 7.19 和定理 8.18 相结合,

^① 考虑到这个术语应用的普遍性,我们在正文中并未按照习惯介绍其出处,这种构造出自参考文献[173, 176]。

并基于 ε 类 (几乎处处) 具有指数电路复杂度的假设, 可得到一个具有指数扩张的正则去随机发生器^①。结合定理 8.16, 可证明以下定理的第一部分。

定理 8.19 (对 BPP 类的去随机化, 重述)

(1) 设 ε 中包含一个判定问题, 电路复杂度几乎处处为指数 (即存在常数 $\varepsilon_0 > 0$, 使得除了有限个 m 外, 任意能在 $\{0,1\}^m$ 上正确判定该问题的电路规模至少为 $2^{\varepsilon_0 m}$), 从而 $BPP = P$ 。

(2) 假设对任意多项式 p , ε 中包含一个判定问题, 其电路复杂度几乎处处大于 p , 则 BPP 包含于 $\bigcap_{\varepsilon > 0} D_{TIME}(t_\varepsilon)$ 中, 其中 $t_\varepsilon(n) \stackrel{\text{def}}{=} 2^{n^\varepsilon}$ 。

第 2 部分可以利用定理 8.18 的推广来证明 (见习题 8.23), 习题 8.22 中给出了该推广形式。注意: 定理 8.19 的第 2 部分取代了定理 8.13 (见习题 7.24)。与通用的伪随机数发生器情况相同, 定理 8.19 的每一部分中的困难性假设都足以证明存在相应的正则去随机发生器 (见习题 8.24)。

定理 8.19 的两部分实际上体现了两种极端情形: 第 1 部分 (通常被认为是 “高端” 部分) 假设了很强的电路下限, 并得到 “完全去随机化” (即 $BPP = P$); 第 2 部分 (通常被认为是 “低端” 部分) 假设了一个极弱的电路下限, 并得到较弱但有意义的去随机化。中间的结论 (依赖于中级的下限假设) 可用类似于习题 8.23 的方法得到, 但是为得到紧凑折中, 采用的方法完全不同 (见参考文献[226])。

8.3.2.2 对构造的分析 (证明定理 8.18)

利用 f 和 S 的时间复杂度上限, G 可在指数时间计算。因此, 这里的重点是证明无法由规模为 $l(k)^2$ 的电路区分 $\{G(U_k)\}$ 与 $\{U_{l(k)}\}$, 具体来讲, G 满足式 (8.11)。事实上, 我们将对 $G'(s) = s \cdot G(s)$ 证明这一点成立, 换言之, 即使是将种子作为辅助输入, G 也可骗过这种电路 (事实上, 这些电路的规模小于 G 的运行时间, 所以无法对给定种子计算 G)。

首先介绍证明思想。作为热身, 假设在构造中使用的集合 (即 $S(k,i)$) 不相交 (然而这是不可能的, 因为 $k < l(k) \cdot m(k)$)。此时, 由 f 的不可近似性易证 $G(U_k)$ 的伪随机性, 因为 G 中包含了将 f 应用于种子不交叠的部分 (见习题 8.19)。在将要分析的实际构造中, 各个集合 (即 $S(k,i)$) 并非不相交, 然而, 两两之间交集很小, 这意味着 G 要将 f 应用于具有较小交叠的种子中。直观上, 这种小交叠保证了 f 对于相应输入的值是 “计算上独立的” (即 f 对某些输入 $x_1, x_2, \dots, x_i$ 的值对于近似计算另一个输入 x_{i+1} 毫无帮助)。为证实这种直觉的正确性, 需要证明: 当固定了所有在目标输入 (即 x_{i+1}) 中不出现的比特时, 之前的函数值 (即 $f(x_1), f(x_2), \dots, f(x_i)$) 可以用较小的计算代价来求出。因此, $f(x_1), f(x_2), \dots, f(x_i)$ 的值并不会 (明显地) 为估算 $f(x_{i+1})$ 提供帮助。有了这个直观印象, 现在来证明定理。

按照惯例, 证明中使用可约性方法, 假设 G' 不能欺骗某个规模为 $l(k)^2$ 的电路, 则得出的结论与谓词 f 是 T -不可近似的假设相矛盾。这种方法利用了伪随机性与不可预测性间的关系 (见 8.2.5 节)。具体而言, 如习题 8.20 中说明的, 任意以 $1/6$ 的缝隙区分 $G'(U_k)$ 与 $U_{l(k)+k}$ 的电路可得到具有类似规模的下一比特预测器, 其成功预测下一比特的概率至少是 $\frac{1}{2} + \frac{1}{6l'(k)} > \frac{1}{2} +$

① 具体地, 从电路复杂度至少为 $\exp(\varepsilon_0 m)$ 的函数出发, 应用定理 7.19, 可对 $T(m) = 2^{\varepsilon m}$ 得到一个 T -不可近似谓词, 其中常数 $\varepsilon \in (0, \varepsilon_0)$ 取决于常数 ε_0 。下面令 $\gamma = \varepsilon/2$ 并利用习题 8.19, 其中确定了 $\alpha, \beta > 0$, 使得 $l(k) = 2^{\alpha k}$ 而 $m(k) = \beta k$ 。注意: (可能需要减小 α) 此时有 $(2\alpha/\beta) + \gamma < \varepsilon$ 。

$\frac{1}{7l(k)}$, 其中因子 $l'(k) = l(k) + k < (1 + o(1))l(k)$ 由混合技术得出 (见式 (8.7))。进一步地, 在当前证明的非一致性环境下, 可以固定要预测的位置 $i+1$, 而不是对任意位置分析预测过程。事实上, $i \geq k$ 一定成立, 因为 $G'(U_k)$ 的前 k 个比特是均匀分布的。在证明的剩余部分, 将上述预测器转化为估算 f 的电路, 该电路比假设中允许的电路要好 (对 f 的不可近似性)。

假设有一个小规模电路 C' 可以预测 $G'(U_k)$ 的第 $i+1$ 比特, 当给定前 i 个比特时, 我们构造一个小规模电路 C 来对输入 $U_{m(k)}$ 估算 $f(U_{m(k)})$ 。关键在于 $G'(s)$ 的第 $i+1$ 比特等于 $f(s_{S(k,j+1)})$, 其中 $j = i - k \geq 0$, 所以 C' 可基于 $s, f(s_{S(k,1)}), f(s_{S(k,2)}), \dots, f(s_{S(k,j)})$ 来估算 $f(s_{S(k,j+1)})$, 其中 $s \in \{0,1\}^k$ 是均匀分布的。注意: 这正是我们的目标, 除了要寻找的电路可能只有 $s_{S(k,j+1)}$ 作为输入以外。

首先要注意当固定对 s 中比特的最佳选择, 且这些位置并不在 $S_{j+1} = S(k, j+1)$ 处 (即选择比特 $s_{[k] \setminus S_{j+1}}$) 时, C' 仍保持其优势。即由平均法, 有

$$\begin{aligned} & \max_{s' \in \{0,1\}^{k-m(k)}} \left\{ \Pr_{s \in \{0,1\}^k} \left[C' \left(s, f(s_{S_1}), f(s_{S_2}), \dots, f(s_{S_j}) \right) = f(s_{S_{j+1}}) \mid s_{[k] \setminus S_{j+1}} = s' \right] \right\} \\ & \geq p' \stackrel{\text{def}}{=} \Pr_{s \in \{0,1\}^k} \left[C' \left(s, f(s_{S_1}), f(s_{S_2}), \dots, f(s_{S_j}) \right) = f(s_{S_{j+1}}) \right] \end{aligned}$$

由假设 $p' > \frac{1}{2} + \frac{1}{7l(k)}$, 将固定串 s' 嵌入到 C' 中, 并令 $\pi(x)$ 表示满足 $s_{S_{j+1}} = x$ 和 $s_{[k] \setminus S_{j+1}} = s'$ 的 (唯一的) 串 s , 可得到电路 C'' , 满足

$$\Pr_{x \in \{0,1\}^m} \left[C'' \left(x, f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j}) \right) = f(x) \right] \geq p'$$

C'' 基本上就是我们要找的电路。唯一的问题是 C'' 的输入不仅仅是 x , 还有 $f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j})$, 而我们要求对 $f(x)$ 的预测器输入仅为 x 。

一个重要的观察是这些“缺失”的值 $f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j})$ 只依赖于 x 中较少比特。这是由于假设对 $t = 1, 2, \dots, j$, 有 $|S_t \cap S_{j+1}| \leq \gamma \cdot m(k)$, 也就是说, $\pi(x)_{S_t}$ 是一个 $m(k)$ -比特串, 其中的 $m_t \stackrel{\text{def}}{=} |S_t \cap S_{j+1}|$ 比特由 x 映射得到, 其余的比特由固定串 s' 映射得到。因此, 给定 x , $f(\pi(x)_{S_t})$ 的值可由规模为 $\tilde{O}(2^{m_t})$ 的 (平凡) 电路求出, 该电路实现 m_t 比特的查表运算。利用所有这些电路 (及 C''), 可得到对于 f 的预测器。具体细节如下。

我们已经得到了目标电路, 记作 C , 其对 f 的 T -近似过程如下。电路 C 依赖于索引 j 和上述分析中的给定串 s' 。回顾 C 集成了计算 $x \mapsto f(\pi(x)_{S_t})$ 的电路 (规模为 $\tilde{O}(2^{\gamma|x|})$), 其中 $t = 1, 2, \dots, j$ 。对于输入 $x \in \{0,1\}^m$, 电路 C 计算 $f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j})$, 并对输入 x 和这些值调用电路 C'' , 输出便是对 $f(x)$ 的猜测, 即

$$C(x) = C'' \left(x, f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j}) \right) = C' \left(\pi(x), f(\pi(x)_{S_1}), f(\pi(x)_{S_2}), \dots, f(\pi(x)_{S_j}) \right)$$

由上述分析, $\Pr_x[C(x)=f(x)] \geq p' > \frac{1}{2} + \frac{1}{7l(k)}$, 其下限为 $\frac{1}{2} + \frac{1}{T(m(k))}$, 由于 $T(m(k)) = 2^{\varepsilon m(k)} = 2^{\varepsilon \beta k} \gg 2^{\alpha k} \gg 7l(k)$, 其中第一个不等式成立是由于 $\varepsilon > 2\alpha/\beta$, 而第二个不等式成立是由于 $l(k) = 2^{\alpha k}$ 。C 的规模上限为 $l(k)^2 + l(k) \cdot \tilde{O}(2^{\gamma \cdot m(k)}) \ll \tilde{O}(l(k)^2 \cdot 2^{\gamma \cdot m(k)}) = \tilde{O}(2^{2\alpha \cdot (m(k)/\beta) + \gamma \cdot m(k)}) \ll T(m(k))$, 而最后一个不等式成立是由于 $T(m(k)) = 2^{\varepsilon m(k)} \gg \tilde{O}(2^{(2\alpha/\beta) \cdot m(k) + \gamma \cdot m(k)})$ (这又利用了 $\varepsilon > (2\alpha/\beta) + \gamma$), 从而得到的结论与 f 为 T -不可近似的假设矛盾。定理 8.18 得证。

8.3.3 技术上的变化及概念思考

本小节先讨论构造 8.17 中体现的通用框架, 最后讨论去随机化的概念。

8.3.3.1 作为通用框架的构造 8.17

Nisan-Wigderson 构造 (构造 8.17) 实际上是一个通用框架, 它有各种不同的实现方法。一些实例是基于对构造方法的抽象及分析而构造的, 以下简要介绍几个这种实例。

首先要注意构造 8.17 中的发生器是一个通用算法, 实现时可以使用任意谓词 f 。进一步地, 这个算法, 记作 G , 实际上是一个预言机, 它向函数 f (非适应地) 发出询问, 因此两者组合起来可以写成 G^f 。类似地, 对 G^f 伪随机性的证明 (即定理 8.18 证明的主要部分) 也是一个通用方案, 对任意 f , 可得到 (非一致的) 电路 C , 给定向 G^f 任意区分器的预言访问时, 该电路可近似计算 f 。电路 C 依赖于 f (以受限方式)。具体地, C 中包含一些查找表, 固定 f 的某些输入比特并求函数值 (即函数 $f(\pi(\cdot)_{s_i})$ 的查找表)。上述抽象使得构造 8.17 的以下实例描述起来更加方便。

常数深度电路的去随机化

构造 8.17 的此种实现利用了奇偶校验函数作为不可近似谓词 f , 注意: 奇偶校验实际上不能由“小的”常数深度电路来估算。在充分的参数环境下, 可以得到扩张为 $l(k) = \exp(k^{1/O(1)})$ 的伪随机数发生器, 能欺骗“小的”常数深度电路 (见参考文献[173])。对这种构造的分析与定理 8.18 的证明极为相似。关键在于将计算 $f(\pi(x)_{s_i})$ 的电路 (直接地) 集成为区分电路时, 电路深度只增加了 2 层。具体地, 电路 C 利用深度为 2 的电路计算 $f(\pi(x)_{s_i})$, 然后对这些值调用一次 (给定的) 区分器, 从而得到对 $f(x)$ 的预测。

最终得到的伪随机数发生器, 利用长为多项式-对数的种子 (等价地, $l(k) = \exp(k^{1/O(1)})$), 可对 $\mathcal{RAC}^0$ (即随机的 $\mathcal{AC}^0$) 去随机化, 类似于定理 8.16。因此, 可在半多项式时间内, 在加法误差内确定地估算满足给定 (常数深度的) 电路的输入片段。具体地, 对任意常数 d , 给定一个深度为 d 的电路 C , 可以确定地估算满足 C 的输入片段 (即使 C 输出为 1 的输入片段), 估算过程可在时间 $\exp((\log|C|)^{O(d)})$ 内完成, 且误差为一个加法常数。^①注意: 即使当 $d=2$ (即 CNF/DNF 范式) 时, 构造确定的多项式时间近似算法也是一个公开问题。

① 我们指出, 在估算 DNF 范式的可满足赋值数量时, 相应的错误估计可通过将一个确定性归约施加于加法常数误差而得到 (见 6.2.2.1 节)。因此, 利用一个能欺骗 DNF 范式的伪随机数发生器, 可以确定地得到对于给定 DNF 范式的可满足性赋值数量的相关 (不一定是加法) 错误估计。

概率证明系统的去随机化

构造 8.17 另一种（且更令人惊奇的）不同实现中，使用由能访问 $\mathcal{NP}$ 预言机的小规模电路无法估算的谓词。结果得到的伪随机数发生器能抵抗一个两阶段的、公开随机数的交互证明（与常数圈交互证明系统功能相同（见 9.1.4.1 节））。关键是从对构造 8.17 的分析中，当给定对区分器的预言访问时，可以得到求相应谓词近似值的黑盒程序（且当区分器为非确定型机器时，该程序仍可发挥作用）。因此，在适当强度（从而仍可信）的假设下，常数圈的交互式证明可以衰减到 $\mathcal{NP}$ 。注意如果偏离上述框架，则可得到一个更强的结论，随后介绍这个结论（见参考文献[167]）。

随机提取器的构造

构造 8.17 一种更自由的实现可用于构造随机提取器（见附录 D.4）。此时的谓词 f 被看作是某个随机函数（的纠错编码），要使构造有意义，必须以黑盒方式使用 f 。在分析中利用一个事实，即可利用相对（关于 f ）较少的信息及对 G^f 的区分器（的黑盒访问）来对 f 求近似。更多细节详见附录 D.4.2.2。

8.3.3.2 对去随机化的思考

定理 8.19 的第 1 部分通常被概括为（在某些合理假设下）“随机性是无用的”。我们认为即使是在传统的复杂性课程中，这一解释也不正确，而在传统课程之外更是大错特错。以下详细阐述。

更深入地观察定理 8.16（其中蕴含了定理 8.19），注意到，一个时间复杂度为 t 的随机算法 A 可由时间复杂度为 $t' = \text{poly}(t)$ 的确定算法 A' 仿真。进一步地，注意到， $A' = A_G$ 在调用 A （以及 G 的正则去随机化） $\Omega(t)$ 次（因为 $l(k) = O(2^k)$ 蕴含了 $2^k = \Omega(t)$ ）之后，可以推断 $t' = \Omega(t)^2$ 必成立。因此，通过定理 8.19（的第 1 部分）去随机化并不是无代价的。

更重要的是，在分布式环境下的去随机化是不可能的，此时，通信双方有利益冲突，因此需要使用随机数来保护自身利益。著名的例子包括许多密码学原语（如加密）以及大多数的概率证明系统（如 PCP）。进一步的讨论可见第 9 章及附录 C。随机性还可用于其他环境中（或在不可能与可能之间，或在无法完成与可以完成之间），包括分布式计算（见参考文献[17]）、通信复杂度（见参考文献[148]）、并行体系结构（见参考文献[151]）、采样（见附录 D.3）以及性能测试（见 10.1.2 节）。

8.4 空间受限的区分器

前两节讨论的发生器，输出序列对于任意高效程序而言是随机的，这里的高效程序定义为时间受限的计算。具体地，8.2 节讨论了多项式时间程序的不可区分性。另一类粒度更细的时间受限程序还需考虑空间复杂度，即对于时间受限的计算，再进一步地规定空间复杂度。这第 5 章的重点，可以据此定义能欺骗空间受限区分器的伪随机数发生器。有趣的是，与 8.2 节和 8.3 节的伪随机数发生器相反，可以不使用任何计算假设来证明能欺骗空间受限区分器的伪随机数发生器的存在性。

预备知识

本小节在技术上是自包含的，其中各种定义可参考 6.1.5.1 节，建议阅读 6.1.5 节作为本

节的背景知识。

8.4.1 定义

研究空间受限区分器的主要动机是在空间受限的随机算法中定义伪随机性。即将这些算法中的随机数用伪随机数（可能基于很短的随机种子生成）替代时，算法仍保持其性质。因此，本节先回顾空间受限的随机算法。然而，空间受限的计算在概念上极为微妙，特别是涉及到非确定性或随机性时（分别见 5.3 节和 6.1.5 节）。对于随机的空间受限计算，需要明确给出两个定义，即时间界限和对随机带的访问类型。

（1）时间界限。对于空间受限的机器，是否要限制其时间复杂度最多为空间复杂度的指数？^①在确定算法中，这样限制很自然（定理 5.3），并且不失一般性，可假设在非确定性情况中也有类似上限（见 5.3.2 节）。但在随机算法中，这个上限不一定成立（见 6.1.5.1 节）。进一步地，由于无法对随机的空间受限算法进行时间复杂度上的限制，不仅使这类算法不太正常，而且无意中使其过长（见 6.1.5.1 节）。

与 6.1.5 节相同，在对随机化的空间受限算法建立模型时，假设其时间复杂度最多为空间复杂度的指数。

（2）对随机带的访问。随机算法的模型中，利用一个特殊的随机带提供随机数。问题在于，空间受限的机器是以单向还是双向（即无限制）方式访问其随机带（允许双向访问意味着随机化可以“无限制地”记录，即不计入空间复杂度（见 5.3 节和 6.1.5 节的讨论））。

对随机带的单向访问对应于在线的随机机器这种自然模型，其行为基于内部随机数而确定（并因此不能“无限制地”保存其以往的随机输入）。因此，与 6.1.5 节一样，我们考虑单向访问模式。^②

综上所述，本节的重点是时间复杂度至多为空间复杂度指数的空间受限随机计算，且以单向方式访问其随机带。

定义上述空间受限随机计算的伪随机性时，要注意在区分器的构造中，主要输入是固定的，随机带被视为区分器的唯一输入。因此，与上述随机化的空间受限计算相一致，考虑这仅受限的区分器，以单向方式访问其要检查的输入序列。下面讨论其中涉及到的算法类型。

区分器是一个空间受限算法，以单向方式访问其输入。算法在每一步基于临时存储内容，或者读取下一个输入比特，或者停留在输入带的当前位置，在两种情况下，都会修改临时存储单元的内容。为简化分析，考虑相应的非一致模型，算法在每一步读取下一个输入比特，并根据一个作用于之前存储内容（以及新的比特）的任意函数来更新临时存储单元。注意这里对模型进行了强化，允许“任意的”（更新）函数，这些函数可用规模为空间界限指数的（非一致性）电路来实现，而不是使用在空间界限的指数时间内（一致性地）计算的（更新）函数。之所以要这样强化，是由于对空间受限的非一致区分器，伪随机数发生器的已知构造仍有效，并且在去随机化过程中总会使用非一致的区分器。

① 作为另一种选择，考虑这些机器是必须停机还是只以接近 1 的概率停机。可以证明，唯一能保证“绝对停机”的方法是令时间复杂度至多为空间复杂度的指数（在目前的讨论中，以及本节全部内容中，我们假设空间复杂度至多为对数）。

② 如果只考虑单向访问，则有助于在无困难性假设的条件下得到空间高效的生成器。而在双向的空间受限计算中，类似生成器的存在蕴含了该领域中有突破性的困难性结论。

上述非一致空间受限算法(或自动机)^①的计算可以表示为有向分层图,其中每层结点对应着临时存储单元可能保存的内容,而相邻层间的迁移对应于一步计算。可以预想使用这种模型描述区分器。在对这些分层图赋予参数时,根据的是相应概率空间(如 $\{G(U_k)\}_{k \in \mathbb{N}}$ 和 $\{U_{l(k)}\}_{k \in \mathbb{N}}$)的索引,记作 k 。换言之,将输入长度,记作 $l=l(k)$,和空间界限,记作 $s(k)$,都表示为参数 k 的函数。这样就将一个非一致的自动空间 $s: \mathbb{N} \leftarrow \mathbb{N}$ 定义为有向分层图中带标记边的一个类, $\{D_k\}_{k \in \mathbb{N}}$,且满足以下条件。

- 有向图 D_k 包含 $l(k)+1$ 层,每层最多有 $2^{s(k)}$ 个结点。第一层有一个结点,是图的(唯一)源结点(无进入边),最后一层包含了图中所有的终点(无外出边)。
- D_k 中的有向边都在相邻层之间,方向是从第 i 层到第 $i+1$ 层, $i \leq l(k)$ 。在标记这些边时,使 D_k 中每个(非终点)结点有两个(可能是平等的)外出有向边,一个标为0另一个标为1。

该自动机对于足够长输入(即长为 l ,而 D_k 有 $l+1$ 层)的计算结果,被定义为用输入中相应比特标记的边序列后最终到达的结点(位于最后一层)。即对于输入 $x = x_1 x_2 \cdots x_l$,在第 i 步($i=1,2,\dots,l$),由当前结点(位于第 i 层)经标记为 x_i 的外出边移动到与其相邻的结点之一(位于第 $i+1$ 层)。利用对最后一层结点的固定划分,可以自然地定义判定的概念(由 D_k),即如果对于输入 x ,自动机 D_k 能到达属于划分中第一部分的结点,则有 $D_k(x)=1$ 。

定义 8.20 (空间受限机的不可区分性)

- 对于非一致自动机 $\{D_k\}_{k \in \mathbb{N}}$ 以及两个概率空间 $\{X_k\}_{k \in \mathbb{N}}$ 和 $\{Y_k\}_{k \in \mathbb{N}}$,如下定义的函数 $d: \mathbb{N} \rightarrow [0,1]$

$$d(k) \stackrel{\text{def}}{=} \left| \Pr[D_k(X_k)=1] - \Pr[D_k(Y_k)=1] \right|$$

称为 $\{D_k\}$ 对两个空间的区分缝隙。

- 设 $s: \mathbb{N} \rightarrow \mathbb{N}$ 和 $\varepsilon: \mathbb{N} \rightarrow [0,1]$ 。概率空间 $\{X_k\}_{k \in \mathbb{N}}$ 称为 (s, ε) -伪随机的,如果对任意空间为 $s(\cdot)$ 的(非一致性)自动机,其对 $\{X_k\}_{k \in \mathbb{N}}$ 和相应的均匀空间(即 $\{U_{l(k)}\}_{k \in \mathbb{N}}$)之间的区分缝隙至多为 $\varepsilon(\cdot)$ 。

- 扩张为 l 的确定算法 G 称为 (s, ε) -伪随机数发生器,如果概率空间 $\{G(U_k)\}_{k \in \mathbb{N}}$ 是 (s, ε) -伪随机的。即任意空间为 $s(\cdot)$ 的非一致性自动机对 $\{G(U_k)\}_{k \in \mathbb{N}}$ 和 $\{U_{l(k)}\}_{k \in \mathbb{N}}$ 的区分缝隙至多为 $\varepsilon(\cdot)$ 。

因此,当使用长为 k 的随机种子时,一个 (s, ε) -伪随机数发生器输出长为 $l(k)$ 的序列,对运行于空间 $s(k)$ 的观察者而言是随机的。注意: $s(k) \leq k$ 是 $(s, 0.5)$ -伪随机数发生器存在的一个必要条件,因为空间为 $s(k) > k$ 的非一致自动机可以识别发生器的像(其中包含至多 2^k 个长为 $l(k) > k$ 的串)。更一般地,在 $s(k)-k$ 与 (s, ε) -伪随机数发生器的扩张之间存在着折中,

^① 使用术语“自动机”(而非算法或机器)是为了提醒读者这种计算设备以单向方式读取输入。还可能使用其他一些术语诸如“实时的”或“在线的”机器。我们建议不使用“在线的”机器,这是为了与随机的(在线)算法区分开,后者可以自由地读取其输入(并以在线方式访问随机源)。事实上,这里的自动机正是由随机算法引出的,算法中固定了基本输入,并将随机源作为其(仅有的)输入。还要注意自动机是有序二元判定表(Ordered Binary Decision Diagrams, OBDDs, 见参考文献[237])的一种特例。

详见习题 8.25 和习题 8.26。

8.4.2 两种构造

与欺骗时间受限区分器的伪随机数发生器相反, 能欺骗空间受限区分器的伪随机数发生器可以不依赖于任何计算假设而构造。以下两个定理表明了区分器的空间界限与发生器扩张函数间折衷的两种极端情况。^①我们强调两个定理都没有像习题 8.26 那样给出参数, 但它们针对着非常高效的构造。首先从扩张的最大化开始。

定理 8.21 (空间界限 $s(k) = \sqrt{k}$ 下的指数扩张) 对任意在给定空间内构造的函数 $s: \mathbf{N} \rightarrow \mathbf{N}$, 存在一个扩张函数为 $l(k) = 2^{k/O(s(k))} \leq 2^{s(k)}$ 的 $(s, 2^{-s})$ -伪随机数发生器。进一步地, 该发生器的工作空间是种子长度的线性函数, 时间则是扩张函数的线性函数。

换言之, 对任意 $t \leq m$, 存在一个发生器, 当输入长为 $k = O(t \cdot m)$ 的随机数种子时, 产生长为 2^t 的伪随机序列, 对任意工作在空间 m 内的 (非一致性) 自动机而言, 该输出序列是随机的 (其区分缝隙至多为 2^{-m})。特别地, 利用长为 $k = O(m^2)$ 的随机数种子, 可以生成长为 2^m 的序列, 对任意工作于空间 m 内的 (非一致性) 自动机, 该序列是随机的。因此, 在任意一个使用随机序列的空间受限计算中, 可将随机序列替换为由种子高效生成的, 周期为空间界限平方的序列。后者的常见实例是对数空间算法。在 8.4.2.2 节中, 利用定理 8.21 (及其思想) 对空间复杂度类如 BPL (即 BPP 类的对数空间对应) 去随机化。定理 8.21 本身在 8.4.2.1 节中证明。

现在转向将区分器的空间界限最大化的情况。注意定理 8.22 只保证了亚指数的区分缝隙 (而非定理 8.21 中的指数区分缝隙)。过去由于忽视这种局限性曾造成一些失误。

定理 8.22 (多项式扩张与线性空间界限) 对任意多项式 p 及某些 $s(k) = k/O(1)$, 存在一个扩张为 p 的 $(s, 2^{-\sqrt{s}})$ -伪随机数发生器。进一步地, 该发生器工作于线性空间和多项式时间 (都用种子长度表示) 内。

换句话说, 存在一个发生器, 输入长为 $k = O(m)$ 的随机种子, 生成长为 $\text{poly}(m)$ 的序列, 该序列对空间为 m 的 (非一致性) 自动机而言是随机的。因此, 可将任意多项式时间和线性空间的随机算法转化为在功能上等价的具有类似时间和空间复杂度的随机运算, 但只使用线性数量的随机数。

8.4.2.1 定理 8.21 和定理 8.22 的证明概要

在证明的一开始, 都考虑一个一般的空间受限区分器, 并证明可以更改该区分器要测试的输入分布 (从均匀分布变为伪随机分布) 而令区分器无法查觉。由这种更改 (实际上是一系列更改) 可以构造一种伪随机数发生器, 在证明的最后部分将详细解释。

定理 8.21 的证明概要^②

证明使用的主要工具是两两独立 Hash 函数的“最大性质”(见附录 D.2)。设函数簇 H_n 为 $\{0,1\}^n$ 到其自身的映射, 称 H_n 是混合的 (Mixing), 如果对任意一对子集 $A, B \subseteq \{0,1\}^n$, 除极少数 (即所占比例为 $\exp(-\Omega(n))$) 的函数 $h \in H_n$ 之外, 有

① 参考文献[12]对这两个结论进行“嫁接”: 存在一个“空间欺骗”的伪随机数发生器参数类, 可将两种极端特例都包含在内。

② 详细证明见参考文献[174]。

$$\Pr[U_n \in A \wedge h(U_n) \in B] \approx \frac{|A|}{2^n} \cdot \frac{|B|}{2^n} \quad (8.13)$$

其中近似项为一个附加的 $\exp(-\Omega(n))$ 项 (见引理 D.4 的推广, 其中暗示可令 $\exp(-\Omega(n))$ 等于 $2^{-n/3}$)。

不失一般性, 可以假设 $s(k) = \Omega(\sqrt{k})$, 则有 $l(k) \leq 2^{s(k)}$ 。对如定义 8.20 中所述的任意 $s(k)$ -空间区分器 D_k , 考虑一个辅助的“区分器” D'_k , 构造方法是“缩短” D_k 中 $n = \Theta(s(k))$ 个连续层中的每一块, 从而得到一个有向分层图, 其中共有 $l' = l(k)/n < 2^{s(k)}$ 层 (且每层有 $2^{s(k)}$ 个结点)。

- D'_k 中每个结点有 2^n 个 (可能平等的) 有向边, 通往下一层中的不同结点。

- 每条有向边由 n 比特串标记, 使得 D'_k 中标记为 $\sigma_1 \sigma_2 \cdots \sigma_n$ 的有向边 (u, v) 代替 D_k 中结点 u 与 v 之间含 n 条边的有向路径, 路径中的边分别被标记为 $\sigma_1, \sigma_2, \dots, \sigma_n$ 。

图 D'_k 显然可以模仿图 D_k , 即 D'_k 对长为 $l(k) = l' \cdot n$ 的输入的计算被定义为将输入分割成长为 n 的连续子串, 并经过由相应 n 比特子串所标记的路径。

关键在于 D'_k 无法区分随机的 $l' \cdot n$ 比特输入 (即 $U_{l',n} \equiv U_n^{(1)} U_n^{(2)} \cdots U_n^{(l')}$) 与形如 $U_n^{(1)} h(U_n^{(1)}) U_n^{(2)} h(U_n^{(2)}) \cdots U_n^{(l'/2)} h(U_n^{(l'/2)})$ 的“伪随机”输入, 其中 $h \in H_n$ 为 (适当选择的) Hash 函数。为证明这一断言, 考虑任意一对相邻结点, u 和 v (分别位于第 i 层和第 $i+1$ 层), 并将所有由 u 到 v 的边的标记集合记作 $L_{u,v} \subseteq \{0,1\}^n$ 。类似地, 对于第 $i+2$ 层的结点 w , 令 $L_{v,w}'$ 表示所有由 v 到 w 的边的标记集合。由式 (8.13) 可知, 除极少数 $h \in H_n$ 之外, 有

$$\Pr[U_n \in L_{u,v} \wedge h(U_n) \in L_{v,w}'] \approx \Pr[U_n \in L_{u,v}] \cdot \Pr[U_n \in L_{v,w}'] \quad (8.14)$$

其中“极少数”和“ $\approx$ ”的含义与式 (8.13) 相同。因此, 对于除占 $\exp(-\Omega(n))$ 比例之外的所有 $h \in H_n$, 将第二次迁移 (由第 $i+1$ 层到第 $i+2$ 层) 中使用的随机数用 h 作用于第一次迁移 (即由第 i 层到第 $i+1$ 层) 中使用的随机数后输出的结果代替, 可以近似地保持 D'_k 由 u 经 v 到达 w 的概率。注意: 利用联合界限 (对所有上述的三元组 (u, v, w)) 可得到, 对于除占 $2^{3s(k)} \cdot l' \cdot \exp(-\Omega(n))$ 比例之外的所有 $h \in H_n$, 上述代替近似地保持了 D'_k 通过 D'_k 中任意一个特定双边路径的概率。

根据 $l' < 2^{s(k)}$ 及适当选择的 $n = \Theta(s(k))$, 有 $2^{3s(k)} \cdot l' \cdot \exp(-\Omega(n)) < \exp(-\Omega(n))$, 因此除“个别” $h \in H_n$ 外, 几乎所有 h 都适用于估算所有这些迁移概率 (我们强调同一个 h 可用于所有估算中)。因此, 以额外的 $|h|$ 个随机比特的代价, 可将 D'_k 中迁移使用的真随机数数量减少一半, 而不会明显影响 D'_k 的最终判定 (其中再次利用 $l' \cdot \exp(-\Omega(n)) < \exp(-\Omega(n))$ 的事实, 这蕴含了近似概率不会增加得太多)。换句话说, 以额外的 $|h|$ 个随机比特的代价, 可以将区分器缩短一半长度, 而大体上保持区分器接受一个随机输入的概率。即固定一个“好的” h (对所有 $2^{3s(k)} \cdot l'$ 个双边路径, 能较好地估算迁移概率), 可将 D'_k 中的双边路径用新的区分器 D_k'' (依赖于 h) 中的路径来代替, 使得标记为 $r \in \{0,1\}^n$ 的边 (u, w) 出现于 D_k'' 中当且仅当对

某个 v , 路径 (u, v, w) 出现于 D'_k 中, 且其第一条边 (即 (u, v)) 标记为 r , 第二条边 (即 (v, w)) 标记为 $h(r)$ 。当然, 关键在于概率 $\Pr[D_k''(U_{(l'/2)n})=1]$ 近似等于 $\Pr[D'_k(U_{l'n})=1]$ 。

可将上述过程应用于 D_k'' , 得到长度缩短一半的区分器 D_k''' , 照此类推。每次可将当前区分器长度缩短一半, 只需随机选择 (并固定) 一个新的 Hash 函数。因此, 重复上述过程对数次 (是 D'_k 长度的对数) 之后, 将得到一个区分器, 只需测试 n 个比特便可停机。在这个过程中一共使用了 $t \stackrel{\text{def}}{=} \log_2(l'/n) < \log_2 l(k)$ 个随机 Hash 函数, 记作 $h^{(1)}, h^{(2)}, \dots, h^{(t)}$ 。这意味着我们可以生成一个 (伪随机) 序列, 使用长为 $n + t \cdot \log_2 |H_n|$ 的种子, 可以欺骗原始区分器 D_k (图 8.3 和习题 8.28)。利用 $n = \Theta(s(k))$ 和充分的函数类 H_n (如构造 D.3), 可以得到期望的 $(s, 2^{-s})$ -伪随机数发生器, 使用的种子长度为 $O(s(k) \cdot \log_2 l(k)) = k$ 。□

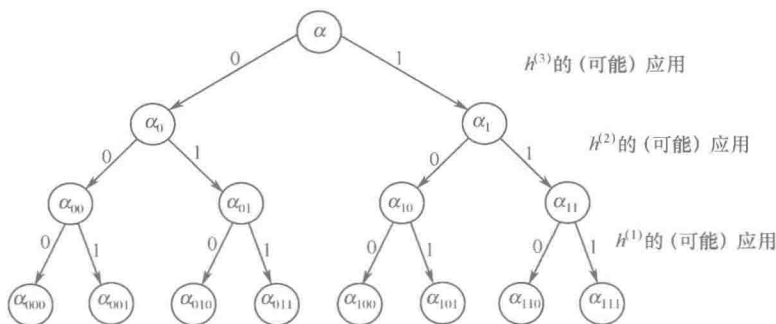


图 8.3 第一个可“欺骗”空间受限自动机 ($i=3$) 的伪随机数发生器。该发生器 (关于种子 $\alpha, h^{(1)}, h^{(2)}, \dots, h^{(t)}$) 的输出包含树的叶子结点中标记为 $\alpha_0, \dots, \alpha_r$ 的串的级联。对任意 $x \in \{0, 1\}^*$, 有 $\alpha_{x_0} = \alpha_x$ 及 $\alpha_{x_1} = h^{(t-|x|)}(\alpha_x)$ 。

特别地, 当 $t=3$ 时, 有 $\alpha_{011} = h^{(1)}(\alpha_{01})$, 这又等于 $h^{(1)}(h^{(2)}(\alpha_0)) = h^{(1)}(h^{(2)}(\alpha))$, 其中 $\alpha = \alpha_\lambda$

定理 8.22 的证明概要^①

证明使用的主要技术是随机性提取器 (定义见附录 D.4.1.1), 这个工具比 Hash 函数更强大。其基本思路是当区分器 D_k 位于某个“较远的”层时, 如第 $t = \Omega(s(k))$ 层, 则典型地“不知道”通往那里的随机选择。即 D_k 只有 $s(k)$ 比特存储空间, 这将使关于以前移动中的 $t-s(k)$ 比特信息为“不确定的”(或随机的)。因此, 许多导致 D_k 到当前状态的随机数也许会被“重用”(或“循环利用”)。为了重用这些随机比特, 需要由上述 $\{0, 1\}^t$ 上熵^②至少为 $t-s(k)$ 的分布上长度足够的串中提取几乎均匀的分布。进一步地, 这个提取过程需要某些额外的真随机比特, 然而, 只需要极少量。为此, 使用 $k' = \Omega(\log t)$ 比特真随机数, 提取出的比特与均匀分布的距离为 $\exp(-\Omega(k'))$ 。

上述过程重复的次数足够多时, 得到的结果将很有意义。为此, 将 k 比特种子分为两部

① 详细证明见参考答案[177]。

② 实际上, 需要为后一种分布增加一个更强的条件: 在第 t 层, 自动机 D_k 以显著优势位于有多于 $2^{0.99(t-s(k))}$ 种方式到达的结点。此时, 该分布表示以大于 $0.99 \cdot (t-s(k))$ 的最小熵到达该结点的随机漫游。推荐读者参考附录 D.4.1.1 中最小熵和提取器的定义。

分, 记作 $r' \in \{0,1\}^{k/2}$ 和 $(r_1, r_2, \dots, r_{3\sqrt{k}})$, 其中 $|r_i| = \sqrt{k}/6$, 并令 $n = k/3$ 。直观上, r' 用于确定前 n 步, 并与 r_i 一起被重用 (或循环利用) 来确定第 $i \cdot n + 1$ 到第 $(i+1) \cdot n$ 步。观察第 $i \cdot n$ 层, 考虑 D_k (到达第 $i \cdot n$ 层某个特定结点时) “已知” 的关于 r' 的信息。假设到达第 $i \cdot n$ 层某个结点时, r' 的条件分布的 (最小) 熵大于 $0.99 \cdot ((k/2) - s(k))$ 。利用 r_i (作为 r' 的提取器的种子), 可以关于 D_k 提取出几乎随机的 (与 U_n 距离为 $2^{-\Omega(\sqrt{k})}$) $0.9 \cdot ((k/2) - s(k) - o(k)) > k/3 = n$ 比特, 并利用这些比特来确定后面的 n 步。因此, 利用 k 个随机比特, 可生成长为 $(1 + 3\sqrt{k}) \cdot n > k^{3/2}$ 的序列, 该序列能欺骗空间界限为 $s(k) = k/10$ 的自动机。具体地, 利用形如 $\text{Ext}: \{0,1\}^{\sqrt{k}/6} \times \{0,1\}^{k/2} \rightarrow \{0,1\}^{k/3}$ 的提取器, 可将种子 $(r', r_1, r_2, \dots, r_{3\sqrt{k}})$ 映射为输出序列 $(r', \text{Ext}(r_1, r'), \text{Ext}(r_2, r'), \dots, \text{Ext}(r_{3\sqrt{k}}, r'))$ 。由此得到一个扩张为 $l(k) = k^{3/2}$ 的 $(s, 2^{-\Omega(\sqrt{s})})$ -伪随机数发生器。

为了使伪随机数发生器的扩张为任意多项式, 而不是某个特定多项式 (即 $l(k) = k^{3/2}$), 我们重复应用一种充分的组合, 以下简要介绍。假设 G_1 为一个具有扩张函数 l_1 的 (s_1, ε_1) -伪随机数发生器, G_2 为具有扩张 l_2 的 (s_1, ε_1) -伪随机数发生器。考虑以下构造的发生器 G 。

(1) 对于输入 $s \in \{0,1\}^k$, 计算 $G_1(s)$, 并将其分成连续的块, 记作 $r_1, r_2, \dots, r_t$, 每块长度为 $k' = s_1(k)/O(1)$, 其中 $t = l_1(k)/k'$ 。

(2) 计算并输出长为 $t \cdot l_2(k')$ 比特的序列 $G_2(r_1), G_2(r_2), \dots, G_2(r_t)$ 。

注意: $|G(s)| = l_1(k) \cdot l_2(k')/k'$, 其中 $k' = s_1(k)/2$, 且 $k = |s|$ 。对于 $s_1(k) = \Theta(k)$, 有 $|G(s)| = l_1(k) \cdot l_2(\Omega(k))/O(k)$, 当 l_1 和 l_2 为多项式时, $|G(s)| = l_1(|s|) \cdot l_2(|s|)/O(|s|)$ 。我们断言, G 为一个 (s, ε) -伪随机数发生器, 其中 $s(k) = \min(s_1(k)/2, s_2(\Omega(s_1(k))))$ 而 $\varepsilon(k) = \varepsilon_1(k) + l_1(k) \cdot \varepsilon_2(\Omega(s_1(k)))$ 。证明使用混合方法, 其中涉及到分布 $G(U_k)$ 和 $U_{t \cdot l_2(k')} \equiv U_{l_2(k')}^{(1)} U_{l_2(k')}^{(2)} \dots U_{l_2(k')}^{(t)}$, 以及混合分布 $I_k \stackrel{\text{def}}{=} G_2(U_k^{(1)}) G_2(U_k^{(2)}) \dots G_2(U_k^{(t)})$ 。 I_k 与 $U_{t \cdot l_2(k')}$ 是 $(s_2(k'), t \cdot \varepsilon_2(k'))$ -不可区分的 (即由空间为 $s_2(k')$ 和区分缝隙为 $t \cdot \varepsilon_1(k')$ 的自动机不可区分), 这一事实可由关于 “多样本不可区分性” (见习题 8.27) 的一般性结论而得到。剩下的工作是要证明 I_k 与 $G(U_k)$ 由空间为 $s_1(k)/2$ 的自动机以缝隙 $\varepsilon_1(k)$ 不可区分。为此, 可将区分器 (对 I_k 和 $G(U_k)$) 转化为对于 $U_{l_1(k)} \equiv U_{l_1(k)}$ 和 $G_1(U_k)$ 的区分器, 其中新的区分器将 $l_1(k)$ 比特输入分为 t 个块 (每块长度为 k'), 再对相应的 k' 比特块调用 G_2 , 并将得到的 $l_1(k')$ 比特序列作为旧区分器的输入。^① □

8.4.2.2 空间复杂性类的去随机化

作为定理 8.21 的直接应用, 可得到 $BPL \subseteq D_{\text{SPACE}}(\log^2)$, 其中 BPL 表示 BPP 类在对

① 新的区分器在读其自身输入的 k' 比特块时保持旧区分器的状态。一旦读取了块 $s' \in \{0,1\}^{k'}$ 后, 新的区分器通过一个相当于旧区分器输入块 $G_2(s')$ 效果的迁移来更新旧区分器的状态。因此, 一个空间为 $s_1(k)/2$ 的区分器可以转化为空间为 $(s_1(k)/2) + k' = s_1(k)$ 的区分器。

数空间中的对应物(见定义 6.11)(回顾 $\mathcal{NL} \subseteq D_{\text{SPACE}}(\log^2)$), 但尚不确定是否 $BPL \subseteq \mathcal{NL}$)。^① 深入分析定理 8.21 的证明过程, 可得到更强的去随机化结论。

定理 8.23 $BPL \subseteq SC$, 其中 SC 表示可由运行于多项式时间和多项式-对数空间的确定算法求解的判定问题。

因此, BPL 类(特别地 $\mathcal{RL} \subseteq BPL$) 是某个不包含 $\mathcal{NL}$ 的类的子类。参考文献[196]中据此得到一个类似结论: 随机对数空间算法可用确定的 $o(\log^2)$ -空间算法模仿; 具体空间为 $\log^{3/2}$ 。最近人们证明了 $\mathcal{RL}$ 中的典型问题属于 $\mathcal{L}$ (见 5.2 节)

定理 8.23 的证明概要^②

下面将利用定理 8.21 证明中构造的发生器, 但是要证明种子的主要部分(即 Hash 函数序列)可以被固定下来(根据区分器), 确定过程可在多项式-对数空间和多项式时间内完成。具体地, 对一个特定的对数空间计算(针对某个特定输入)去随机化时, 先得到相应的区分器(是该对数空间算法的内部随机数的函数), 记作 D_k' 。注意: 某个特定 Hash 函数 $h \in H_n$ 对于具体的 D_k' 是否有良好定义, 这一问题可在 $n = |h|/2$ 的线性空间及 D_k' 规模的对数空间内判定。这个判定过程的时间复杂度是其空间复杂度的指数。从而, 对于给定 D_k' , 可以在这样的复杂度范围内找到一个良好的 $h \in H_n$ (扫描所有可能的 $h \in H_n$)。一旦找到了良好的 h , 也可以在同样复杂度内构造出相应的图 D_k'' (其中的边表示 D_k' 中含两条边的路径)。实际上, 更有意义的是, 可以通过 D_k' 的两步(即两次边遍历)来确定 D_k'' 的一步(一次边遍历)。这样就可以对 D_k'' 固定一个 Hash 函数, 以此类推。具体细节如下。

定理的主要结论是: 寻找长为 $t \stackrel{\text{def}}{=} \log_2 l'(k)$ 的良好 Hash 函数序列的过程可在空间 $t \cdot O(n + \log |D_k|) = O(n + \log |D_k|)^2$ 和时间 $\text{poly}(2^n \cdot |D_k|)$ 内完成。即时间复杂度是空间复杂度的亚指数函数(时间复杂度远远小于界限 $\exp(O(n + \log |D_k|)^2)$)。从 $D_k^{(1)} = D_k'$ 开始, (对于 $D_k^{(1)}$) 找到一个良好的 Hash 函数 $h^{(1)} \in H_n$, 这就确定了 $D_k^{(2)} = D_k''$ 。在找到(并保存) $h^{(1)}, h^{(2)}, \dots, h^{(i)}$ 之后, 就确定了 $D_k^{(i+1)}$ 。通过模仿 $D_k^{(i+1)}$ 中的边遍历对, 可得到关于 $D_k^{(i+1)}$ 的良好 Hash 函数 $h^{(i+1)} \in H_n$ 。关键在于并没有构造图序列 $D_k^{(2)}, D_k^{(3)}, \dots, D_k^{(i+1)}$, 而是通过在 D_k' 中进行 2^i 次边遍历来模仿 $D_k^{(i+1)}$ 的一次边遍历。利用 $h^{(1)}, h^{(2)}, \dots, h^{(i)}$, 从 $D_k^{(i+1)}$ 中的结点 v 开始的(边遍历中的)移动 $\alpha \in \{0, 1\}^n$, 转化为 D_k' 中由 v 开始的 2^i 个移动序列, 其中的移动由以下 2^i -长的(n 比特串)序列确定:

$$\bar{h}^{(0^i)}(\alpha), \bar{h}^{(0^{i-2}01)}(\alpha), \bar{h}^{(0^{i-2}10)}(\alpha), \bar{h}^{(0^{i-2}11)}(\alpha), \dots, \bar{h}^{(1^i)}(\alpha)$$

其中 $\bar{h}^{(\sigma_i \dots \sigma_2 \sigma_1)}$ 是对由 $\sigma_i \dots \sigma_2 \sigma_1$ 确定的函数 $h^{(i)}, \dots, h^{(2)}, h^{(1)}$ 中一部分进行合成后而得到的。具体地, $\bar{h}^{(\sigma_i \dots \sigma_2 \sigma_1)}$ 等于 $h^{(i_{i'})} \circ \dots \circ h^{(i_2)} \circ h^{(i_1)}$, 其中 $i_1 < i_2 < \dots < i_{i'}$, 而 $\{i_j : j = 1, 2, \dots, i'\} = \{j : \sigma_j = 1\}$ 。

如果能实现 $D_k^{(i+1)}$ 中的边遍历, 则可以确定一个给定函数 $h \in H_n$ 对于 $D_k^{(i+1)}$ 是否良好。方

① 事实上, $\mathcal{RP}$ 类的对数空间对应物, 记作 $\mathcal{RL}$, 包含于 $\mathcal{NL} \subseteq D_{\text{SPACE}}(\log^2)$ 中, 因此, 定理 8.21 暗示了 $\mathcal{RL} \subseteq D_{\text{SPACE}}(\log^2)$ 是无意义的。

② 证明详见参考文献[175]。

法是考虑 $D_k^{(i+1)}$ 中所有相关的三元组 (u, v, w) , 对每个 (u, v, w) 计算式 (8.14) 中的 3 个量 (即概率), 并作出相应的判定。尝试所有可能的 $h \in H_n$, 可以找到对 $D_k^{(i+1)}$ 而言为良好的函数 (记作 $h^{(i+1)}$)。这个过程利用额外的存储空间 $s' = O\left(n + \log\left|D_k'\right|\right)$ (在用于保存 $h^{(1)}, h^{(2)}, \dots, h^{(i)}$ 的空间之外), 其时间复杂度为 s' 的指数。因此, 对于给定 D_k' , 可在 s' 的指数时间和空间 $s' + t \cdot \log_2 |H_n| = O(t \cdot s')$ 内找到良好的 Hash 函数序列 $h^{(1)}, h^{(2)}, \dots, h^{(t)}$ 。这个序列可以模仿 $D_k^{(t+1)}$ 中的边遍历, 后者又可用于 (确定地) 估算 D_k' 接受一个随机输入的概率 (即由位于第一层的一个源结点出发, 自动机 D_k' 到达最后一层某个接受结点的概率)。在估算时, 可以遍历 $D_k^{(t+1)}$ 中所有 2^n 条边, 并求相应概率。

总之, 给定 D_k' , 可以在 $O(t \cdot s')$ 空间和 $\exp(O(s' + n))$ 时间内 (确定地) 估算 D_k' 接受一个随机输入的概率, 其中 $s' = O\left(n + \log\left|D_k'\right|\right)$, $t < \log_2 |D_k'|$ 。对于 $n = \Theta\left(\log\left|D_k'\right|\right)$, 这意味着 $O\left(\log\left|D_k'\right|\right)^2$ -空间和 $\text{poly}\left(\left|D_k'\right|\right)$ -时间。我们指出为使这种估算更精确, 最多只需附加一个 $1/\text{poly}\left(\left|D_k'\right|\right)$ 项, 在这里加上 $1/6$ 即可。

最后, 回顾这种估算与 BPL 类的去随机化之间的联系 (事实上, 注意定理 8.13 证明中的类比)。对数空间概率机 M 对输入 x 的计算过程可以表示为一个规模为 $\text{poly}(|x|)$ 的有向分层图 $G_{M,x}$ 。具体地, 图中每一层的结点代表 $M(x)$ 计算中可能的配置, 而位于第 i 层和第 $i+1$ 层之间的边表示计算的第 i 次迁移, 它依赖于 M 的随机带上第 i 比特 (或等价地, 依赖于 M 的第 i 个随机输入)。^①因此, M 接受 x 的概率等于由 $G_{M,x}$ 中第一层的一个结点出发, 经随机漫游到达最后一层某个结点的概率, 最后一层的结点代表被接受的配置。令 $k = \Theta(\log|x|)$ 和 $n = \Theta(k)$, 图 $G_{M,x}$ 与定理 8.21 证明一开始提到的图 D_k 相同, 而可以通过对 D_k 进行 (如上的) “ n 层” 提取来得到 D_k' 。进一步地, D_k 与 D_k' 可在对数空间内 (由 x) 构造 (并利用引理 5.2 中的模仿计算, 其中将 D_k' 作为输入)。结合上述分析, 可以得到结论, M 接受 x 的概率可在 $O(\log|x|)^2$ -空间和 $\text{poly}(|x|)$ -时间内近似确定。定理得证。□

8.5 特殊用途的伪随机数发生器

目前为止, 我们讨论的伪随机数发生器都可以减少算法使用的随机数 (甚至是对相应复杂性类完全地去随机化)。例如, 我们讨论了对 BPP 类和 BPL 类的去随机化。本节的要求则没有那么高。只希望对特定算法或以某种 (受限) 方式使用随机比特的算法类去随机化 (或减少其随机性)。例如, 算法可能只要求其中的随机数序列 (或该序列中的 “块”) 是两两独立的。这里的限制条件具有具体而 “结构化的” 形式, 而不像前几节那样是抽象的复杂性理论形式。

^① 注意: $G_{M,x}$ 是图的 “分层版本”, 在定理 5.11 的证明中提到过 (记作 G_x)。进一步地, 定理 5.11 的证明中, 考虑某个路径的存在性, 在这里则考虑其数量 (或遍历其中之一的概率)。

上述限制引出严格受限的相应区分器类，它们比前几节的区分器类更弱。由这些受限区分器又可以得到较弱类型的伪随机数发生器（生成序列可欺骗这些区分器）。此类发生器在复杂性理论（及上文所暗示的算法中）也有许多应用（只给出简介）。

首先介绍一种最简单的发生器：两两独立发生器，并对 $t \geq 2$ ，任意推广到 t 元独立的情况。这类发生器可以完美地骗过只能观察到输出序列中 t 个位置的区分器。由此很自然地得到几乎两两（或 t 元）独立的发生器，也能欺骗上述区分器（虽然不是完全欺骗）。后一种发生器蕴含于一类更强的发生器中：小偏移发生器，这类发生器本身也很有意义。小偏移发生器可欺骗任意线性测试（即任意只涉及在输入序列某些固定位置进行异或运算的区分器）。然后讨论扩张随机行走发生器（Expander Random Walk Generator）：这种发生器生成的串与任意密度的子串相同的概率接近于真随机序列与子串相同的概率。相关概念如采样器、分离器（Disperser）和提取器的定义可参考附录 D。

教学注释：与前几节的构造不同，本节提出的构造不要求对计算类型（时间或空间受限）有深入理解。相反，这些构造均基于各种纯数学方法，对其分析则出现在本章习题中。

参数说明

为了与前几节保持一致，本节继续用种子长度（记作 k ）来表示发生器。但在本节大部分结论中，这并非通用的表达方式，所以在脚注中提供了通用表达，其中种子长度由其他参数确定。

8.5.1 两两独立发生器

两两（或 t -元）独立发生器可以欺骗只检查输出序列中两个（或 t 个）元素的测试。这些本地测试实际上是严格受限的，然而，在许多环境中这很正常。例如，测试对应的（对程序的）概率分析只依赖于程序做出的两两独立选择。而在某些自然的参数范围内，两两独立的采样与完全独立的采样一样好，见附录 D.1.2 和附录 D.3。

一个块长度为 $b: \mathbf{N} \rightarrow \mathbf{N}$ （以及扩张函数 l ）的 t -元独立发生器是一个相对高效的确定算法（工作于输出长度的多项式时间），可将 k 比特随机种子扩张为 $l(k)/b(k)$ 个块构成的序列，每块长度为 $b(k)$ ，满足任意 t 个块是均匀的且在 $\{0,1\}^{tb(k)}$ 上独立分布。将发生器（对于种子 s ）输出的第 i 块记作 $G(s)_i$ ，要求对任意 $i_1 < i_2 < \dots < i_t$ （在 $[l(k)/b(k)]$ 中），有

$$G(U_k)_{i_1}, G(U_k)_{i_2}, \dots, G(U_k)_{i_t} \equiv U_{t \cdot b(k)} \quad (8.15)$$

注意：即使当 t 个块为适应性选择时，上述条件也成立（见习题 8.29）。当 $t=2$ 时，称发生器为两两独立的。

8.5.1.1 构造

第一种构造中，考虑 $\text{GF}(2^{b(k)})$ ，即含 $2^{b(k)}$ 个元素的有限域，并将其中元素与 $\{0,1\}^{b(k)}$ 相关联。

命题 8.24 (t -元独立发生器)^① 设 t 为给定整数， $b, l, l': \mathbf{N} \rightarrow \mathbf{N}$ 满足 $b(k) = k/t$ ，

^① 在这种 t -元独立发生器的一般表示中，种子长度被看作是相应块长度和扩张的函数。即给定参数 b 和 $l' \leq 2^b$ ，种子长度被设置为 $t \cdot b$ 。

$l'(k) = l(k)/b(k) > t$ 及 $l'(k) \leq 2^{b(k)}$ 。令 $\alpha_1, \alpha_2, \dots, \alpha_{l'(k)}$ 为域 $\text{GF}(2^{b(k)})$ 中固定的不同元素。对于 $s_0, s_1, \dots, s_{t-1} \in \{0, 1\}^{b(k)}$, 令

$$G(s_0, s_1, \dots, s_{t-1}) \stackrel{\text{def}}{=} \left(\sum_{j=0}^{t-1} s_j \alpha_1^j, \sum_{j=0}^{t-1} s_j \alpha_2^j, \dots, \sum_{j=0}^{t-1} s_j \alpha_{l'(k)}^j \right) \quad (8.16)$$

其中运算在 $\text{GF}(2^{b(k)})$ 上进行, 则 G 是一个 t -元独立的, 具有块长度 b 和扩张 l 的发生器。

也就是说, 给定一个包含 $\text{GF}(2^{b(k)})$ 中 t 个元素的种子, 发生器输出含 $l'(k)$ 个元素的序列。

命题 8.24 的证明留作习题 (习题 8.30)。其基础是: 对任意固定的 $v_0, v_1, \dots, v_{t-1}$, 条件 $\left\{ G(s_0, s_1, \dots, s_{t-1})_{i_j} = v_j \right\}_{j=1}^t$ 构成了含 t 个方程的 $\text{GF}(2^{b(k)})$ 上的线性方程组 (关于变量 $s_0, s_1, \dots, s_{t-1}$), 且这些方程线性无关 (因此, 从某个表达式的线性无关性可以得到相应随机变量的统计无关性)。

一个稍显冗长的说明

我们指出, 式 (8.16) 并没有给出 (发生器的) 一种明确构造。其中缺乏对 $\text{GF}(2^{b(k)})$ 的明确描述, 为此, 需要一个 $\text{GF}(2)$ 上的 $b(k)$ 次不可约多项式。对于特定的 $b(k)$ 值, 存在一种良好的描述, 如当 $d \stackrel{\text{def}}{=} b(k) = 2 \cdot 3^e$ (e 为整数) 时, 多项式 $x^d + x^{d/2} + 1$ 在 $\text{GF}(2)$ 上不可约。进一步的细节见附录 G.3。

注意: 类似于式 (8.16) 的另一种构造可在任意有限域上实现 (如含任意素数幂个元素的有限域), 但是要明确表示这种域并不容易 (如求大小近似为 $2^{b(k)}$ 的素数个数)。这使人们转而研究 $t=2$ 的情形。

以下构造利用了 (随机的) 仿射变换 (作为可能的种子)。事实上, 利用由 Toeplitz 矩阵确定的仿射变换, 可以达到更好的性能 (即更短的种子)。所谓 Toeplitz 矩阵是指对角线元素均相同的矩阵 (图 8.4), 即矩阵 $T = (t_{i,j})$ 为 Toeplitz 矩阵, 如果对所有 i, j , 有 $t_{i,j} = t_{i+1,j+1}$, 注意: 一个 Toeplitz 矩阵可用其第一行和第一列的元素确定 (即所有 $t_{1,j}$ 和 $t_{i,1}$)。

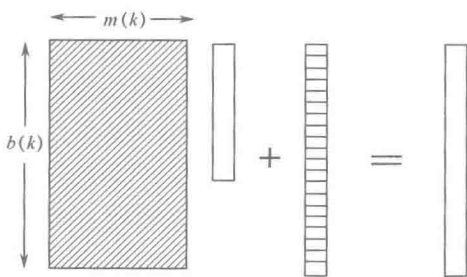


图 8.4 Toeplitz 矩阵确定的仿射变换

命题 8.25 (另一种两两独立发生器, 见图 8.4)^① 令 $b, l, l', m: \mathbf{N} \rightarrow \mathbf{N}$ 满足 $l'(k) = l(k)/b(k)$

^① 在这种两两独立发生器的一般性描述中, 种子的长度是期望块长度和扩张的函数。即给定参数 b 和 l' , 种子长度被设为 $2b + \lceil \log_2 l' \rceil - 1$ 。

及 $m(k) = \lceil \log_2 l'(k) \rceil = k - 2b(k) + 1$, 将 $\{0,1\}^n$ 与 $\text{GF}(2)$ 上的 n 维向量空间相关联, 并令 $v_1, v_2, \dots, v_{l'(k)}$ 为 $\text{GF}(2)$ 上 $m(k)$ 维向量空间中的不同向量。对于 $s \in \{0,1\}^{b(k)+m(k)-1}$ 和 $r \in \{0,1\}^{b(k)}$, 令

$$G(s, r) \stackrel{\text{def}}{=} (T_s v_1 + r, T_s v_2 + r, \dots, T_s v_{l'(k)} + r) \quad (8.17)$$

其中 T_s 为由串 s 定义的 $b(k)$ 行 $m(k)$ 列 Toeplitz 矩阵, 则 G 是具有块长度 b 和扩张 l 的两两独立发生器。

给定一个种子, 输入由 $b(k)$ 行 $m(k)$ 列 Toeplitz 矩阵和 $b(k)$ 维向量所定义的仿射变换, 则发生器输出 $l'(k) \leq 2^{m(k)}$ 个串构成的序列, 每个串的长度为 $b(k)$ 。注意: $k = 2b(k) + m(k) - 1$, 且扩张性质要求 $l'(k) > k/b(k)$ 。命题 8.25 的证明留作习题 (习题 8.31)。证明还利用了这样一个现象: 由某个表达式的线性无关性可得到相应随机变量的统计无关性, 这里 $\{G(s, r)_{i_j} = v_j\}_{j=1}^2$ 为 $\text{GF}(2)$ 上含 $2b(k)$ 个方程的线性方程组 (关于代表 s 和 r 中比特的布尔变量), 且这些方程线性无关。我们指出, 类似于式 (8.17) 的构造可在任意有限域上实现。

更强的高效生成

忽略较大有限域的表示问题, 如果发生器可在其长度的多项式时间内生成输出, 则上述两种构造是高效的。实际上, 上述构造均满足更强的高效生成概念, 在某些应用中这一概念非常有用。具体地, 存在一个多项式时间算法, 给定种子 $s \in \{0,1\}^k$ 及块位置 $i \in [l'(k)]$ (二进制形式), 可输出相应输出中的第 i 块 (即 $G(s)$ 的第 i 块)。注意: 在第一种构造中 (式 (8.16)), 更强意味着在 $\text{poly}(k)$ -时间内求有限域 $\text{GF}(2^{b(k)})$ 的表示^①。当 $b(k)$ 具有形式 $2 \cdot 3^e$ 时, 这是可能的。

8.5.1.2 应用 (简介)

两两独立发生器有许多应用 (见参考文献 [161, 238])。大部分应用基于这样一个事实: “大数定律”在两两独立 (而非完全独立) 测试序列中成立。Chebyshev 不等式 (见附录 D.1.2.2) 体现了这一点, 它是附录 D.3 中采样 (典型) 应用的基础。作为一个具体实例, 以下讨论对解最大独立集问题 (见参考文献 [168] 中 12.3 节) 的快速并行算法的去随机化。^②一般来说, 对随机算法的分析只需假设某些对象是两两独立的, 从而可以用两两独立发生器的输出序列作为算法中的随机数。因此, 两两独立发生器足以欺骗由随机算法所构造的区分器。

由式 (8.16) 看出, 对任意常数 $t \geq 2$, 去随机化 (搜索所有 2^k 个可能的种子) 的代价是块长度的指数 (因为 $b(k) = k/t$)。另一方面, 块的数量最多为块长度的指数 (因为 $l'(k) \leq 2^{b(k)}$), 所以如果需要大量的块, 则可以人为地增加块长度 (即令 $b(k) = \log_2 l'(k)$)。因此, 去随机化的代价是 $\max(l'(k), 2^{b'(k)})$ 的多项式, 其中 $l'(k)$ 表示所需要的块数量, 而 $b'(k)$ 是期望的块长度 (换言之, $l'(k)$ 表示期望的随机选择数量, $2^{b'(k)}$ 表示每个这种选择的域规模)。

① 要达到基本的高效性, 在 $\text{poly}(l(k))$ -时间内找到 $\text{GF}(2^{b(k)})$ 的表示就足够了, 当 $b(k) = O(\log(l(k)))$ 时, 这可以通过穷举搜索实现。

② 算法的核心是选择每个结点的概率与结点数成反比。分析时, 则只要求这些选择是两两独立的, 在一个足够大的集合中均匀取值即可满足要求。

从而, 如果对随机算法的分析建立于许多随机选择间常数元独立性之上, 且每个选项规模合理时, 去随机化是可行的。

8.5.2 小偏移发生器

如 8.5.1.2 节所述, $O(1)$ -元独立发生器可以高效地对任意高效随机算法去随机化, 对其分析仅基于随机带上常数数量比特的独立性。这是由于具有扩张 l 的 t -元独立发生器需要的种子长度为 $\Omega(t \cdot \log l)$ 。构造小偏移发生器的初始动机就是尝试让去随机化过程具有超出常数的独立性 (使用的种子长度是伪随机序列长度的对数)。具体地, 8.5.2.2 节中将提到, 从小偏移发生器出发, 能近似生成 t -元独立序列 (基于对数长度种子)。

上述去随机化是小偏移发生器的一种重要应用, 后者也有独立意义, 可用于许多其他场合。特别是小偏移发生器可欺骗检查所有输出序列的“全局测试”而非检查输出中固定数量的位置 (如有限独立发生器的情形)。具体而言, 由小偏移发生器生成的序列可欺骗任意的线性测试 (即测试中需计算序列比特的线性组合)。

对于 $\varepsilon: \mathbb{N} \rightarrow [0, 1]$, 扩张为 l 的 ε -偏移发生器是一个相对高效的确定算法 (工作于 $\text{poly}(l(k))$ 时间内), 将 k 比特随机种子扩张为 $l(k)$ 比特的序列, 使得对任意给定的非空集合 $S \subseteq \{1, 2, \dots, l(k)\}$, 输出序列在 S 上的偏移最多为 $\varepsilon(k)$ 。 n 个 (可能独立的) 随机布尔变量 $\zeta_1, \zeta_2, \dots, \zeta_n \in \{0, 1\}$ 序列在集合 $S \subseteq \{1, 2, \dots, n\}$ 上的偏移定义为

$$2 \cdot \left| \Pr[\oplus_{i \in S} \zeta_i = 1] - \frac{1}{2} \right| = \left| \Pr[\oplus_{i \in S} \zeta_i = 1] - \Pr[\oplus_{i \in S} \zeta_i = 0] \right| \quad (8.18)$$

这里引入因子 2 是为了让这些偏移对应于分布的傅立叶系数 (被视为从 $\{0, 1\}^n$ 到实数的函数)。为了显示这种相关性, 用 $\{\pm 1\}$ 来代替 $\{0, 1\}$, 并用乘法代替 XOR 运算, 则关于集合 S 的偏移可以写成

$$\left| \Pr \left[\prod_{i \in S} \zeta_i = +1 \right] - \Pr \left[\prod_{i \in S} \zeta_i = -1 \right] \right| = \left| E \left[\prod_{i \in S} \zeta_i \right] \right| \quad (8.19)$$

这恰好就是关于 S 的傅里叶系数 (的绝对值)。

8.5.2.1 构造

已知一种具指数扩张和指数衰减偏移的相对高效的小偏移发生器。

定理 8.26 (小偏移发生器)^① 对某些通用的常数 $c > 0$, 令 $l: \mathbb{N} \rightarrow \mathbb{N}$ 和 $\varepsilon: \mathbb{N} \rightarrow [0, 1]$ 满足 $l(k) \leq \varepsilon(k) \cdot \exp(k/c)$, 则存在一个扩张为 l 的 ε -偏移发生器, 其运行时间为输出长度的多项式。

特别地, 可令 $l(k) = \exp(k/2c)$ 及 $\varepsilon(k) = \exp(-k/2c)$ 。已知有 3 种满足定理 8.26 的小偏移发生器 (见参考文献[10])。其中之一基于线性反馈移位寄存器 (LFSR) 构造, 发生器的种子用于确定 LFSR 的“反馈逻辑”和“初始状态”。具体地, 一个 t 阶 LFSR 的反馈逻辑是 $\text{GF}(2)$ 上的 t 次不可约多项式, 记作 $f(x) = x^t + \sum_{j=0}^{t-1} f_j x^j$, 其中 $f_0 = 1$, 而 LFSR 由初态 $s_0 s_1 \dots s_{t-1} \in \{0, 1\}^t$ 的所生成的 (l 比特长) 输出序列记作 $r_0 r_1 \dots r_{l-1}$, 其中 (图 8.5)

① 在这种发生器的一般性描述中, 种子长度是期望偏移和扩张的函数, 即给定参数 ε 和 l , 则种子长度为 $c \cdot \log(l/\varepsilon)$ 。我们指出, 利用参考文献[10], 常数 c 仅为 2 (即 $k \approx 2 \log_2(l/\varepsilon)$), 而利用参考文献[170], $k \approx \log_2 l + 4 \log_2(1/\varepsilon)$ 。

$$r_i = \begin{cases} s_i & , i \in \{0, 1, \dots, t-1\} \\ \sum_{j=0}^{t-1} f_j \cdot r_{i-t+j} & , i \in \{t, t+1, \dots, l-1\} \end{cases} \quad (8.20)$$

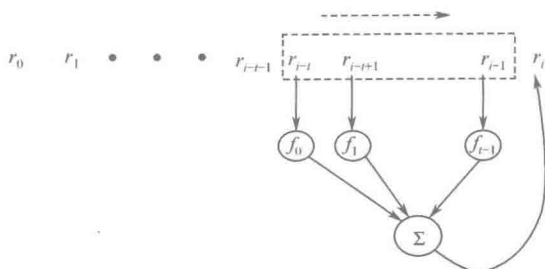


图 8.5 小偏移 LFSR 发生器 ($t = k/2$)

如前所述, 在相应的小偏移发生器中, k 比特种子用于选择一个几乎均匀分布的反馈逻辑 f (即一个随机的 $t = k/2$ 次不可约多项式) 及一个均匀分布的初始状态 s (即随机 t 比特串)^①。相应的 $l(k)$ 比特输出 $r = r_0 r_1 \cdots r_{l(k)-1}$ 由式 (8.20) 计算。

更强的高效生成

与 8.5.1.1 节相同, 注意上述构造满足更强的高效生成概念。即存在一个多项式时间算法, 给定 k 比特的种子及比特位置 $i \in [l(k)]$ (二进制形式), 算法输出相应输出序列的第 i 比特。具体地, 在 LFSR 的构造中, 给定种子 $f_0, f_1, \dots, f_{(k/2)-1}, s_0, s_1, \dots, s_{(k/2)-1}$ 和比特位置 $i \in [l(k)]$ (二进制形式), 算法输出相应输出中第 i 比特 (即 r_i)。^②

8.5.2.2 应用 (简介)

小偏移发生器的典型应用是生成字符串的短的随机“指纹”(或“摘要”), 可将串间的相等/不相等关系 (概率地) 映射为指纹的相等/不相等。其中的关键是检查 $x = y$ 是否成立可以概率归约为检查 x 与 r 的内积模 2 是否等于 y 与 r 的内积模 2, r 由小偏移发生器 G 生成。因此, 设 s 为 G 的随机种子, v 为 z 与 $G(s)$ 的内积模 2, 对 (s, v) 就是串 z 的随机指纹。这种归约的一个优点是, 检查时只需要在 x 与 y 之间“传输”很少的几个比特 (即发生器的种子及内积) (见习题 8.33)。另一个优点是归约的低随机复杂度, 其中使用 $|s|$ 个, 而非 $|G(s)|$ 个随机比特, 这里的 $|s|$ 可能是 $O(\log |G(s)|)$ 。这种低 (即对数的) 随机复杂度使小偏移发生器可用于构造 PCP 系统 (如可见 9.3.2.2 节), 也可用于设计方程组可满足性问题的缝隙放大归约 (见 9.3.3 节和习题 10.6)。

① 注意: 这种发生器的一种实现要求有一个算法来选择几乎随机的不可约多项式, 次数为 $t = \Omega(k)$ 。一种简单算法是枚举所有的 t 次不可约多项式, 再从中随机选择。该算法可在 $\exp(t)$ 时间内实现 (使用 t 个随机比特), 当 $l(k) = \exp(\Omega(k))$ 时, 时间为 $\text{poly}(l(k))$ 。参考文献[10]的第 8 节描述了一种使用 $O(t)$ 个随机比特的 $\text{poly}(t)$ -时间算法。

② 这一断言基于如下事实

$$\begin{pmatrix} r_{i-t+1} \\ r_{i-t+2} \\ \vdots \\ r_{i-1} \\ r_i \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ f_0 & f_1 & f_2 & \cdots & f_{t-1} \end{pmatrix} \begin{pmatrix} r_{i-t} \\ r_{i-t+1} \\ \vdots \\ r_{i-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ f_0 & f_1 & f_2 & \cdots & f_{t-1} \end{pmatrix}^{i-t+1} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{t-2} \\ s_{t-1} \end{pmatrix}$$

小偏移发生器在许多领域得到了应用（如不可近似性、结构复杂性及应用密码学，见参考文献[90]中 3.6.2 节）。另外，小偏移发生器在各种类型“伪随机”对象的设计中也是一个重要工具。

近似独立的发生器

本节在一开始暗示了，小偏移与有限独立性的近似版本有关。^①实际上，习题 8.34 表明，即使是受限的 ε -偏移量（其中只有含 $t(k)$ 个元素的子集具有偏移上限 ε ）也蕴含了目标序列中任意 $t(k)$ 比特与 $U_{t(k)}$ 是 $2^{t(k)/2} \cdot \varepsilon(k)$ -接近的，这里涉及到两种分布间的变异距离（即 1 阶范数距离）（距离的最大范数上界为 $\varepsilon(k)$ ）。^②结合定理 8.26 与上述上界，可得到具有指数扩张（即 $l(k) = \exp(\Omega(k))$ ）的发生器，当输出序列中任意 $t(k) = \Omega(k)$ 比特与 $U_{t(k)}$ 为 $2^{-\Omega(k)}$ -接近时，输出序列近似于 $\Omega(k)$ 元独立。进一步地，如习题 8.40，根据命题 8.24 中构造的线性性质，可以得到具有双指数扩张的（即 $l(k) = \exp(2^{\Omega(k)})$ ）、近似 $t(k)$ -独立的发生器。也就是说，得到的发生器扩张为 $l(k) = 2^{2^{\Omega(k)}}$ ，其生成的比特序列中，任意 $t(k) = \Omega(k)$ 个位置与均匀分布的变异距离至多为 $\varepsilon(k) = 2^{-\Omega(k)}$ 。换言之，这种发生器种子长度可能是 $k = O(t(k) + \log(1/\varepsilon(k)) + \log \log l(k))$ 。在最大范数距离的相应结论中，只需 $k = O(\log(t(k)/\varepsilon(k)) + \log \log l(k))$ 。因此，只要能基于足够多的二元随机数间的对数元（几乎）独立性来分析随机算法，便可实现去随机化（利用种子长度为对数发生器）。

其他工作将上述构造扩展到非二进制情况（见参考文献[86]3.6.2 节）。某些工作还考虑了与之相关的在几何和组合矩形中构造小“差集”的问题。

t -通用集合发生器

利用上述最大范数的上界（在任意 t 个位置与均匀分布的偏移），当 $\varepsilon < 2^{-t}$ 时，可由任意 ε -偏移发生器得到一个 t -通用集合发生器。后者的输出序列中，任意长为 t 的子序列的所有 2^t 种模式都出现（即每个子序列至少出现在一个种子产生的输出中）。这种发生器也得到了大量应用。

8.5.2.3 小结

本节简要介绍将小偏移发生器推广到输出序列属于任意有限域的情况。对于素特征域 $\text{GF}(p)$ ，首先定义偏移的概念。对式(8.15)进行推广，定义含 n 个随机变量 $\zeta_1, \zeta_2, \dots, \zeta_n \in \text{GF}(p)$ （可能相互独立）的序列关于线性组合 $(c_1, c_2, \dots, c_n) \in \text{GF}(p)^n$ 的偏移为 $\left\| \mathbb{E} \left[\omega^{\sum_{i=1}^n c_i \zeta_i} \right] \right\|$ ，其中 ω 表示 p 次单位（复）根（即当 $p=2$ 时， $\omega=-1$ ）。参考习题 8.42，由偏移 $\zeta_1, \zeta_2, \dots, \zeta_n$ （针对任意非零线性组合）的上界可得到 $\text{GF}(p)$ 上 $\sum_{i=1}^n c_i \zeta_i$ 与均匀分布的距离上界。

称 $S \subseteq \text{GF}(p)^n$ 是一个 ε -偏移的概率空间，如果 S 中一个均匀选择的序列关于 $\text{GF}(p)$ 上任意非零线性组合的偏移最多为 ε （无论该空间是否能高效构造，都能得到相应的 ε -偏移发生器）。我们指出，8.5.2.1 节及习题 8.31 中的 LFSR 构造，实际上可推广到 $\text{GF}(p)$ ，并得到一

① 要注意的是，与完全独立不同，这里只涉及到在固定比特位置的分布。见习题 8.32 中的进一步讨论。

② 两个界限都由差分向量（即两个概率向量之差）的 2 阶范数限得出。细节见习题 8.34。

个规模(最多)为 p^{2^e} 的 ε -偏移概率空间, 其中 $e = \lceil \log_p(n/\varepsilon) \rceil$ 。这种构造可用于推广 8.5.2.2 节中的应用。

8.5.3 扩张图上的随机漫游

本节介绍通过在较大图上进行漫游来生成输出序列的发生器, 其中图的度数较小但有充分的“混合”性质, 被称为扩张图, 通过在其上的随机漫游可得到结点集上的 l' 值序列。这里使用的随机种子长度为 $b + (l' - 1) \cdot \log_2 d$, 其中 2^b 表示图中结点个数, d 为其度数。种子长度可以与 $l' \cdot b$ 个随机比特相比, 后者用于生成 $\{0, 1\}^b$ 上的 l' 个独立采样(或在规模为 2^b 的图上随机漫游)。有趣的是, 在与 $\{0, 1\}^b$ 中任意固定子集产生碰撞方面, (由扩张图中的随机漫游生成的)伪随机序列与真随机序列是非常相似的。首先定义这个性质(或定义相应的碰撞问题)。

定义 8.27 (碰撞问题) $\{0, 1\}^b$ 上(可能独立的)随机变量序列, 记作 $(X_1, X_2, \dots, X_{l'})$, 是 (ε, δ) -碰撞的, 如果对任意元素个数不少于 $\varepsilon \cdot 2^b$ 的(目标)集合 $T \subseteq \{0, 1\}^b$, 至少一个变量与 T 产生碰撞的概率不小于 $1 - \delta$, 即 $\Pr[\exists i, \text{使 } X_i \in T] \geq 1 - \delta$ 。

显然, $\{0, 1\}^b$ 上一个长为 l' 的真随机序列是 (ε, δ) -碰撞的, 其中 $\delta = (1 - \varepsilon)^{l'}$ 。上述的“扩张图随机漫游发生器”(以下将详细描述)有类似性质。具体地, 对任意足够小的 $c > 0$ (依赖于扩张图的度数及混合性质), 以及 $\delta = (1 - (1 - c) \cdot \varepsilon)^{l'}$, 发生器的输出是 (ε, δ) -碰撞的。为描述这种发生器, 需要进一步讨论扩张图。

扩张图

度数为 d 且特征值 $\lambda < d$ 的扩张图(或扩张)是一个无限的 d -正则图(d -regular)集合, $\{G_N\}_{N \in S} (S \subseteq \mathbf{N})$, 其中 G_N 为 N 结点的 d -正则图, 且 G_N 所有特征值的绝对值(除最大的一个外)上限为 λ 。为了简化问题, 假设 G_N 的结点集为 $[N]$ (虽然在某些构造中, 使用更丰富的表示可能更方便)。把这类图称为 (S) 的 (d, λ) -扩张图。这个定义与上述的“混淆”概念相关(混淆性质是指从某个固定结点出发进行随机漫游, 最终到达的结点在图上呈均匀分布的比率)。进一步的细节可参考附录 E.2.1。

我们感兴趣的是这种图的具体构造, 即是否存在一个多项式时间算法, 对于输入 N (二进制形式), G_N 中的结点 v 及索引 $i \in \{1, 2, \dots, d\}$, 返回 v 的第 i 个邻居(还要求使 G_N 存在的集合 S 足够“易处理”——即给定任意 $n \in \mathbf{N}$, 可以高效地找到 $s \in S$, 满足 $n \leq s < 2n$)。已知有几种明确的扩张图构造(见附录 E.2.2)。以下介绍扩张图的相关结论, 其中利用了一个事实: 对任意 $\bar{\lambda} > 0$, 存在 d 及 $\{2^b : b \in \mathbf{N}\}$ 上一种明确的 $(d, \bar{\lambda} \cdot d)$ -扩张图。^①

定理 8.28 (扩张图随机漫游定理) 令 $G = (V, E)$ 为具有度数 d 和特征值界限 λ 的扩张图。 W 为 V 的子集且 $\rho \stackrel{\text{def}}{=} |W|/|V|$, 考虑由 G 上一个均匀选择的结点出发且含 $l' - 1$ 步的漫游, 每一步从与当前结点关联的 d 条边中均匀选择一条, 并经该条边走到下一个结点。此漫游经过 W 的概率至多为

① 当 $d = \text{poly}(1/\bar{\lambda}^2)$ 时有此结论。事实上, $d = O(1/\bar{\lambda}^2)$ 为最优值, 即使图的规模近似为 2 的幂时, 也能得到这个最优值。

$$\rho \cdot \left(\rho + (1-\rho) \cdot \frac{\lambda}{d} \right)^{l'-1} \quad (8.21)$$

因此, 扩张图上的随机漫游关于碰撞性是“伪随机”的(当考虑与集合 $V \setminus W$ 碰撞, 并利用 $\varepsilon = 1 - \rho$ 时), 换言之, 密度为 ε 的集合以 $1 - \delta$ 的概率产生碰撞, 其中 $\delta = (1 - \varepsilon) \cdot (1 - \varepsilon + (\lambda/d) \cdot \varepsilon)^{l'-1} < (1 - (1 - (\lambda/d)) \cdot \varepsilon)^{l'}$ 。参考文献[135]中给出了定理 8.28 的证明, 习题 8.43 中则介绍了比式(8.21)更弱的上限证明概要。利用定理 8.28 及明确的 $(2', \bar{\lambda} \cdot 2')$ -扩张图, 可得到一个伪随机数发生器, 其输出序列对于几乎最优的 δ 是 (ε, δ) -碰撞的。

命题 8.29 (扩张图随机漫游发生器) ^①

• 对任意常数 $\bar{\lambda} > 0$ 及 $\{2^n, n \in \mathbb{N}\}$, 考虑一个明确构造的 $(2', \bar{\lambda} \cdot 2')$ -扩张图, 其中 $t \in \mathbb{N}$ 是一个足够大的常量。对于 $v \in [2^n] \equiv \{0, 1\}^n$ 和 $i \in [2'] \equiv \{0, 1\}^{t'}$, 在相应的 2^n -结点图中, 将由结点 v 出发, 经过其第 i 条边可以到达的结点记作 $\Gamma_i(v)$ 。

• 对于 $b, l': \mathbb{N} \rightarrow \mathbb{N}$, 满足 $k = b(k) + (l'(k) - 1) \cdot t < l'(k) \cdot b(k)$, 以及 $v_0 \in \{0, 1\}^{b(k)}$ 和 $i_1, i_2, \dots, i_{l'(k)-1} \in [2']$, 令

$$G(v_0, i_1, i_2, \dots, i_{l'(k)-1}) \stackrel{\text{def}}{=} (v_0, v_1, v_2, \dots, v_{l'(k)-1}) \quad (8.22)$$

其中

$$v_j = \Gamma_{i_j}(v_{j-1})$$

则 G 的扩张为 $l(k) = l'(k) \cdot b(k)$, 且对任意 $\varepsilon > 0$ 和 $\delta = (1 - (1 - \bar{\lambda}) \cdot \varepsilon)^{l'(k)}$, $G(U_k)$ 是 (ε, δ) -碰撞的。

G 的扩张的最大值为 $b(k) \approx k/2$ (且 $l'(k) = k/2t$), 但是所有应用都不把追求最大扩张作为一个目标。许多应用中参数 n 、 ε 和 δ 是给定的, 目标是构造一个发生器, 输出 $\{0, 1\}^n$ 上 (ε, δ) -碰撞的序列, 同时使序列长度及发生器使用的随机数(即种子长度)个数达到最小。事实上, 命题 8.29 建议使用长为 $l' \approx \varepsilon^{-1} \log_2(1/\delta)$ 的序列, 且由长为 $n + O(l')$ 的随机种子生成。

扩张图随机漫游发生器在许多领域得到应用(如 PCP 和不可近似性(见参考文献[29]中 11.1 节)、密码学(见参考文献[91]中 2.6 节), 以及设计各种类型的“伪随机”对象(特别是附录 D.3))。

本章注释

表 8.1 描述了本章讨论的一些伪随机数发生器。我们强调通用的伪随机数发生器(见 8.2 节)与其他情况下的发生器(见 8.3 节和 8.4 节)是有区别的: 前者的区分器比发生器复杂, 而后者发生器比区分器更复杂。具体来讲, 在通用情况中, 发生器运行于(某个固定的)多项式时间, 并需要经受任意概率多项式时间区分器的攻击。事实上, 8.2 节的某些证明使用了这一事实, 即区分器可以任选种子并调用发生器。相反, 定理 8.18 分析的 Nisan-Widgerson

^① 在这种发生器的一般表示中, 种子长度为期望块长度和扩张的函数, 即给定参数 b 和 l' , 种子长度为 $b + O(l' - 1)$ 。

发生器（见 8.3 节），其运行时间则多于区分器的运行时间，该定理的证明本质上是根据这一事实。类似地，8.4 节介绍的弹性空间发生器的空间复杂度也要高于区分器的空间上限。

表 8.1 伪随机数发生器一览表。其中 OW 表示单向函数存在的假设。EvEC 表示 ε 类具有（几乎处处的）指数电路复杂度的假设

类型	区分器的资源	发生器的资源	扩张（即 $l(k)$ ）	备注
通用的伪随机数发生器	$p(k)$ -时间, p 为任意多项式	$\text{poly}(k)$ -时间	$\text{poly}(k)$	OW
正则去随机发生器	$2^{k/O(1)}$ -时间	$2^{O(k)}$ -时间	$2^{k/O(1)}$	EvEC
空间受限发生器 健壮性	$s(k)$ -空间, $s(k) < k$	$O(k)$ -空间	$2^{k/O(s(k))}$	运行于 $\text{poly}(k) \cdot l(k)$ -时间
	$k/O(1)$ -空间	$O(k)$ -空间	$\text{poly}(k)$	
t -元独立发生器	检查 t 个位置	$\text{poly}(k) \cdot l(k)$ -时间	$2^{k/O(t)}$	（如两两独立）
小偏移发生器	线性测试	$\text{poly}(k) \cdot l(k)$ -时间	$2^{k/O(1)} \cdot \varepsilon(k)$	
扩张图随机漫游发生器	“碰撞”	$\text{poly}(k) \cdot l(k)$ -时间	$l'(k) \cdot b(k)$	
	与 $\{0,1\}^{b(k)}$ 碰撞为 $\left(0.5, 2^{-r(k)/O(1)}\right)$, 其中 $l'(k) = ((k - b(k))/O(1)) + 1$			

伪随机数发生器的通用模式

可将各种不同的伪随机数发生器视为通用模式的实例化。要注意的是，虽然历史上对各种概念的研究在技术层面上看似无关，然而，通用的伪随机数发生器是其他类型发生器的根源。特别地，诸如计算不可区分性、困难性与伪随机性之间的联系，以及伪随机性和不可预测性的等价性，这些概念的第一次问世都是在通用的伪随机数发生器中（并启发人们设计“用于去随机化的发生器”，以及“空间受限的发生器”）。而对特殊用途发生器的研究（见 8.5 节）与上述这些概念没有什么关系。

一般用途的伪随机数发生器

计算不可区分性的概念体现了用计算的方法研究随机性，它由 Goldwasser 和 Micali^[108]在安全加密方案的定义中提出。计算不可区分性的概念在密码学中十分重要（见附录 C）。其一般性定义由 Yao^[239]给出。利用参考文献[108]中的混合技术，Yao 还观察到生成与均匀分布不可区分序列的伪随机数发生器，与生成不可预测序列的发生器在定义上是等价的。后一个定义来自于 Blum 和 Micali^[41]的早期工作。

Blum 和 Micali^[41]是研究伪随机数发生器的先驱，他们基于一些简单的难解性假设来构造伪随机数发生器。例如，构造了基于素域上离散对数问题困难性的伪随机数发生器。其工作中构造了一种基本模式，后续所有的改进型工作均采用该模式（如可见于参考文献[118, 239]）。我们感兴趣的是计算困难性向伪随机性的转化，困难核心谓词（参考文献[41]中也给出了定义），以及迭代方法（见式（8.10））。

定理 8.11（伪随机数发生器存在当且仅当单向函数存在）是 Håstad、Impagliazzo、Levin 和 Luby^[118]的研究成果，其基础是参考文献[99]中的困难核心谓词（见定理 7.7）。然而，定理 8.11 已知的证明方法相当复杂，不适于在本书中介绍。寻找该定理简单而紧凑（见 8.2.7.1 节）的证明是一个非常重要的研究课题。

Goldreich、Goldwasser 和 Micali^[95]提出了伪随机函数（附录 C.3.3 中进一步讨论）的定义，并给出了构造方法。我们还要提到（并推荐阅读）参考文献[96]中对于伪随机对象一般

理论的研究。最后,参考文献[91]中第3章更详细地介绍了通用的伪随机数发生器。

对时间复杂性类的去随机化

Yao^[239]还观察到,可以利用非一致的强伪随机数发生器对时间复杂性类去随机化。Nisan^[173,176]指出了一个重要现象,无论是否以这种方式使用伪随机数发生器,要求发生器的运行时间为其种子长度的指数就足够了,从而发生器的运行时间多于区分器(代表被去随机化的算法)。这一现象引出正则去随机化的定义,以及Nisan和Wigderson构造^[173,176],这是参考文献[128]中进一步改进的基础。定理8.19的第一部分(即对BPP类的“高端”去随机化)由Impagliazzo和Wigderson^[128]证明,而第二部分(即“低端”部分)来自参考文献[176]。

Nisan-Wigderson发生器^[176]后来的应用超出了其原始描述。这些应用包括欺骗非确定型机器(从而对常数圈交互式证明系统去随机化)以及构造随机性提取器^[222](见附录D.4.2.2中的综述)。

上述的去随机化结论均基于某种非一致性(最坏情况)假设,将BPP类置于某种最坏情况下确定型的复杂性类中。与之相反,我们提出一个结论,基于均匀的复杂性(最坏情况)假设,将BPP类置于平均情况下确定型的复杂性类中(见10.2节)。特别是要介绍一个由Impagliazzo和Wigderson^[129]证明的定理(但是没有出现在正文中),其中断言:如果BPP类(几乎处处)不包含于EXP中,则BPP类存在确定的亚指数时间算法,在所有典型情况下都是正确的(即针对任意多项式时间采样的分布)。

针对空间受限区分器的伪随机性

关于“弹性空间伪随机数发生器”的第一篇论文^[4]中指出,^①该研究方向的开辟是由于使用通用伪随机数发生器而得到的去随机化结论。这些结论(必要地)依赖于困难性假设,所以其目标是识别自然的算法类,对这类算法的去随机化不依赖于困难性假设(但是依赖于相应区分器类的已知困难性结论)。这个目标之前在常数深度(随机性)电路中已经实现,但是空间受限的(随机)算法更具吸引力。一些工作给出了弹性空间伪随机数发生器的几种不同构造,但是被8.4.2节提出的两个不可比较的结论所取代:定理8.21(亦称Nisan发生器^[174])和定理8.22(又称Nisan-Zuckerman发生器^[177])。这两个结论在参考文献[12]中被“嫁接”起来。定理8.23($BPL \subseteq SC$)由Nisan证明^[175]。

特殊用途的发生器

8.5节构造的各种发生器并未建立于任意一种其他类型的伪随机数发生器基础上(甚至也没有使用伪随机性的一般概念)。参考文献[54]中明确提出了两两独立发生器(在参考文献[50]中有所暗示)。参考文献[6]中将其推广到 t -元独立($t \geq 2$)的发生器。小偏移发生器最早由Naor和Naor^[170]定义并构造,后来参考文献[10]中给出了3种简单构造。扩张图随机漫游发生器则由Ajtai、Komlos和Szemerédi提出^[4],他们发现扩张图上的随机漫游近似等于多次独立尝试与任意固定的具有足够密度子集(在结点集内)的碰撞。对漫游的碰撞性质分析后来得到改善,达到了定理8.28中的上限,该定理来自于参考文献[135]中引理6.1。

(上述发展历史的注记中没有提到在该领域发挥着重要作用的几个技术性贡献。读者要进一步了解进一步的细节,可以见参考文献[86]中第3章)。事实上,本章是参考文献[90]中第3章的修订版,为主题内容提供了大量细节,并删去了第二手材料(这些材料在修改后收入本书附录D.3和附录D.4)。

① 该文最早提出了扩张图随机漫游技术,故在相关文献中被频繁引用。

参考文献[229]提出了对伪随机性的另一种研究,其中更多地强调各种技术间的关系。特别强调了信息论与计算现象间的联系(即随机提取器和正则去随机化),而本书则尝试将二者分离(可见 8.3 节和附录 D.4)。

习 题

8.1 证明如果对随机数发生器不做任何计算上的要求,则可以无条件地得到关于能欺骗任意亚指数规模电路簇的“发生器”的结论。也就是说,在无计算假设时,证明存在函数 $G:\{0,1\}^* \rightarrow \{0,1\}^*$, 满足 $\{G(U_k)\}_{k \in \mathbb{N}}$ 是(强)伪随机的,其中对任意 $s \in \{0,1\}^*$, 有 $|G(s)| = 2|s|$ 。进一步地,证明 G 可在双指数时间内计算。

提示: 利用概率方法(见参考文献[11])。首先对任意固定的电路 $C:\{0,1\}^n \rightarrow \{0,1\}$, 规定一个概率上限,对于含 $2^{n/2}$ 个元素的随机集合 $S \subset \{0,1\}^n$, $\Pr[C(U_n)=1] - \left|\left\{\{x \in S: C(x)=1\}\right\}/|S|\right|$ 的绝对值大于 $2^{-n/8}$ 。然后利用联合界限,证明存在含 $2^{n/2}$ 个元素的集合 $S \subset \{0,1\}^n$, 使得不存在规模为 $2^{n/5}$ 的电路可以区分 S 中均匀分布的元素与 $\{0,1\}^n$ 上均匀分布的元素,这里区分缝隙为 $2^{-n/8}$ 。

8.2 证明命题 8.3 的如下推论。

(1) 设 A 为概率多项式时间算法,可解判定问题 $\chi:\{0,1\}^* \rightarrow \{0,1\}$ (在 BPP 中),并令 A_G 如构造 8.2 所述。证明不可能找到一个输入 x , 使 A_G 的错误概率远远大于 A 的错误概率,即证明对于输入 1^n , 不可能找到一个 $x \in \{0,1\}^n$, 使得 $\Pr[A_G(x) \neq \chi(x)] < \Pr[A(x) = \chi(x)] + 0.01$ 。

(2) 设 A 为概率多项式时间算法,解与 NP -关系 R 相关的搜索问题,令 A_G 如构造 8.2 所述。证明,不可能找到一个输入 x , 使 A_G 输出错误解。为简单起见,假设 A 的错误概率为 $1/3$, 证明对于输入 1^n , 不可能找到一个 $x \in \{0,1\}^n \cap S_R$, 使得 $\Pr[(x, A_G(x)) \notin R] > 0.4$, 其中 $S_R \stackrel{\text{def}}{=} \{x: \exists y(x, y) \in R\}$ 。类似地,不可能找到一个 $x \in \{0,1\}^n \setminus S_R$, 使得 $\Pr[A_G(x) \neq \perp] > 0.4$ 。

8.3 证明省略式(8.4)中的绝对值符号对于定义 8.4 没有影响。

提示: 考虑 $D'(z) \stackrel{\text{def}}{=} 1 - D(z)$ 。

8.4 证明计算不可区分性是一个等价关系(定义于概率空间对上)。具体地,证明这种关系具有传递性(即 $X \equiv Y$ 且 $Y \equiv Z$ 蕴含着 $X \equiv Z$)。

8.5 证明如果 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 是计算不可区分的, A 为概率多项式时间算法,则 $\{A(X_n)\}_{n \in \mathbb{N}}$ 和 $\{A(Y_n)\}_{n \in \mathbb{N}}$ 也是计算不可区分的。

提示: 如果算法 D 可以区分后者,则满足 $D'(z) \stackrel{\text{def}}{=} D(A(z))$ 的 D' 可以区分前者。

8.6 与习题 8.5 相反,证明当 A 无计算界限时,该结论不一定成立。也就是说,证明存在计算不可区分的概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$ 及指数时间算法 A , 使得 $\{A(X_n)\}_{n \in \mathbb{N}}$ 和 $\{A(Y_n)\}_{n \in \mathbb{N}}$ 并非计算不可区分。

提示: 对任意一对概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$, 考虑布尔函数 f , 满足 $f(z)=1$ 当且仅当

$\Pr[X_n = z] > \Pr[Y_n = z]$ 。证明 $|\Pr[f(X_n) = 1] - \Pr[f(Y_n) = 1]|$ 等于 X_n 与 Y_n 的统计差。考虑对 f 的充分(近似)实现(如求 $\Pr[X_n = z]$ 和 $\Pr[Y_n = z]$ 的近似值, 一直到 $\pm 2^{-2|z|}$)。

8.7 证明伪随机数发生器的存在性蕴含着存在多项式时间构造的概率空间, 这些空间统计距离较远, 因而是计算不可区分的。

提示: 对 $G(U_k)$ 与 $U_{l(k)}$ 的统计距离求下限, 其中 G 是扩张为 l 的伪随机数发生器。

8.8 根据定理 7.7, 对如下事实提供自包含的证明: 存在单向的双射蕴含着存在多项式时间构造的概率空间, 这些空间的统计距离较远, 因而是计算不可区分的。

提示: 设 b 为函数 f 的困难核心, 考虑空间 $\{f(U_n) \cdot b(U_n)\}_{n \in \mathbb{N}}$ 和 $\{f(U_n) \cdot U_1\}_{n \in \mathbb{N}}$ 。利用命题 8.9 的证明思想, 证明这些空间是计算不可区分的。证明如果 f 为双射, 则这些空间统计距离较远。

8.9 (根据参考文献[88]) 证明习题 8.7 中的充分条件实际上也是必要条件。回顾概率空间距离较远的定义, 称 $\{X_n\}_{n \in \mathbb{N}}$ 与 $\{Y_n\}_{n \in \mathbb{N}}$ 统计距离较远, 如果对某个正多项式 p 及所有足够大的 n , X_n 与 Y_n 的变异距离大于 $1/p(n)$ 。利用如下 3 个步骤证明, 如果存在多项式时间构造的概率空间, 统计距离较远从而是计算不可区分的, 则伪随机数发生器存在。

(1) 证明不失一般性, 可以假设 X_n 与 Y_n 的变异距离大于 $1 - \exp(-n)$ 。

提示: 对于上述的 X_n 与 Y_n , 考虑 $\overline{X_n} = (X_n^{(1)}, X_n^{(2)}, \dots, X_n^{(t(n))})$ 及 $\overline{Y_n} = (Y_n^{(1)}, Y_n^{(2)}, \dots, Y_n^{(t(n))})$, 其中 $X_n^{(i)}$ (或 $Y_n^{(i)}$) 为 X_n (或 Y_n) 的独立副本, 且 $t(n) = O(n \cdot p(n)^2)$ 。为了求 $\overline{X_n}$ 与 $\overline{Y_n}$ 的统计差下限, 考虑集合 $S_n = \{z : \Pr[X_n = z] > \Pr[Y_n = z]\}$, 以及表示 $\overline{X_n}$ 在 S_n 中副本数量的随机变量。

(2) 利用第一步中的 $\{X_n\}_{n \in \mathbb{N}}$ 和 $\{Y_n\}_{n \in \mathbb{N}}$, 证明存在一个错误熵发生器 (False Entropy Generator), 它是一个确定的多项式时间算法 G , 满足 $G(U_k)$ 的熵为 $e(k)$, 而 $\{G(U_k)\}_{k \in \mathbb{N}}$ 与多项式时间构造的熵大于 $e(\cdot) + (1/2)$ 的概率空间计算不可区分。

提示: 令 S_0 和 S_1 为采样算法, 满足 $X_n \equiv S_0(U_{\text{poly}(n)})$ 及 $Y_n \equiv S_1(U_{\text{poly}(n)})$ 。考虑发生器 $G(\sigma, r) = (\sigma, S_\sigma(r))$ 及分布 Z_n , 后者以概率 $1/2$ 等于 (U_1, X_n) , 否则, 等于 (U_1, Y_n) 。注意: 在 $G(U_1, U_{\text{poly}(n)})$ 中, 第一个比特几乎可以由其余的比特确定, 而在 Z_n 中, 第一个比特与其余比特统计独立。

(3) 利用一个错误熵发生器构造另一个错误熵发生器, 其超出的熵为 $\sqrt{k}$, 并用后者构造一个伪随机数发生器。

提示: 利用 8.2.5.3 节中的思想 (即定理 8.11 证明中对有用方向的讨论)。

8.10 (多个采样与单个采样的区分) 与命题 8.6 相反, 证明存在两个概率空间, 由单个采样在计算上不可区分, 但是利用两个采样可以高效地区分。进一步地, 这两个空间之一为均匀空间, 而另一个具有稀疏支集 (即只有 $\text{poly}(n)$ 个串被赋予非零概率)。事实上, 第二个空间不能在多项式时间构造。

提示: 证明对任意函数 $d: \{0, 1\}^n \rightarrow [0, 1]$, ($\{0, 1\}^n$ 中) 存在两个串 x_n 和 y_n , 以及 $p \in [0, 1]$, 使得 $\Pr[d(U_n) = 1] = p \cdot \Pr[d(x_n) = 1] + (1 - p) \cdot \Pr[d(y_n) = 1]$ 。将这个断言推广到 m 个函数,

利用 $m+1$ 个串及相应概率的凸组合。^①最后得到结论, 存在分布 Z_n , 具有势至多为 $m+1$ 的支集, 使得对任意 (以词典序排列的) 前 m 个 (随机) 算法 A , 有 $\Pr[A(U_n)=1]=\Pr[A(Z_n)=1]$ 。注意: Z_n 的两个独立采样被赋予同一个值的概率至少是 $1/(m+1)$, 从而得到一个简单的双采样区分器, 可以区分 U_n 与 Z_n 。

8.11 (扩张函数的放大, 另一种构造) 对于构造 8.7 中的 G_l 及 l , 考虑 $G(s) \stackrel{\text{def}}{=} G_l^{l(|s|)-|s|}(s)$, 其中 $G_l^i(x)$ 表示 G_l 对于输入 x 迭代 i 次 (即 $G_l^i(x)=G_l^{i-1}(G_l(x))$ 及 $G_l^0(x)=x$)。证明 G 是一个扩张为 l 的伪随机数发生器。仔细考虑构造 8.7 与这种构造相比的优势 (如考虑运行时间)。

提示: 利用混合方法, 令第 i 次混合为 $G_l^i(U_{l(k)-i})$, $i=0,1,\dots,l(k)-k$ 。注意: $G_l^{i+1}(U_{l(k)-(i+1)})=G_l^i(G_l(U_{l(k)-i-1}))$ 且 $G_l^i(U_{l(k)-i})=G_l^i\left(U_{\left|G_l(U_{l(k)-i-1})\right|}\right)$, 并利用习题 8.5。

8.12 (伪随机性与不可预测性) 证明概率空间 $\{Z_k\}_{k \in \mathbb{N}}$ 是伪随机的当且仅当其是不可预测的。为简单起见, 称 $\{Z_k\}_{k \in \mathbb{N}}$ 是 (下一比特) 不可预测的, 如果对任意概率多项式时间算法 A , $\Pr[A(F_i(Z_k))=B_{i+1}(Z_k)]-(1/2)$ 可以忽略, 其中 $i \in \{0,1,\dots,|Z_k|-1\}$ 为均匀分布, 而 $F_i(z)$ (或 $B_{i+1}(z)$) 表示 z 的 i 比特前缀 (或第 $i+1$ 比特)。

提示: 证明伪随机性蕴含着多项式时间不可预测性, 也就是说, 多项式时间可预测性与伪随机性相悖 (因为如果不考虑计算能力的话, 均匀空间是不可预测的)。利用混合方法证明不可预测性蕴含着伪随机性。具体地, 第 i 个混合包含 Z_k 的 i 比特前缀, 续以 $|Z_k|-i$ 个均匀分布的比特。因此, 如果能区分两端的混合 (对应于 Z_k 与 $U_{|Z_k|}$), 则蕴含着能区分一对随机选择的相邻混合, 这又蕴含了下一比特的可预测性。在最后一步中, 可以利用命题 8.9 的证明方法。

8.13 证明一个概率空间是不可预测的 (从左向或) 当且仅当其是从右向左不可预测的 (或以任意有规律的顺序)。

提示: 利用习题 8.12, 并注意到一个空间是伪随机的当且仅当其逆是伪随机的。

8.14 令 f 为保持长度的双射, b 为 f 的困难核心谓词。对任意多项式 l , 令 $G'(s) \stackrel{\text{def}}{=} b(f^{l(|s|)-1}(s)) \cdots b(f(s)) \cdot b(s)$, 证明 $\{G'(U_k)\}$ 是不可预测的 (在习题 8.12 的意义下)。

提示: 用反证法, 假设对于均匀分布的 $j \in \{0,1,\dots,l(k)-1\}$, 给定 $\{G'(U_k)\}$ 的 j 比特前缀时, 算法 A' 可以预测 $G'(U_k)$ 的第 $j+1$ 比特。也就是说, 给定 $b(f^{l(k)-1}(s)) \cdots b(f^{l(k)-j}(s))$ 时, 算法 A' 可以预测 $b(f^{l(k)-(j+1)}(s))$, 其中 s 在 $\{0,1\}^k$ 上均匀分布。考虑算法 A , 当给定 $y=f(x)$ 时, 通过对输入 $b(f^{j-1}(y)) \cdots b(y)$ 调用 A' 可以近似计算 $b(x)$, 其中 j 在 $\{0,1,\dots,l(k)-1\}$ 中均匀选择。由于 f 诱导出 $\{0,1\}^n$ 上的置换, 从而 $b(f^j(U_k)) \cdots b(f(U_k)) \cdot b(U_k)$ 与 $b(f^{l(k)-1}(U_k)) \cdots b(f^{l(k)-j}(U_k)) \cdot b(f^{l(k)-(j+1)}(U_k))$ 分布相同, 利用这一事实分析 A 的成功概率。

① 也就是说, 证明对任意 m 个函数 $d_1, d_2, \dots, d_m: \{0,1\}^n \rightarrow [0,1]$, 存在 $m+1$ 个串 $z_n^{(1)}, z_n^{(2)}, \dots, z_n^{(m+1)}$ 及 $m+1$ 个和为 1 的非负数字 $p_1, p_2, \dots, p_{m+1}$, 使得对任意 $i \in [m]$, 有 $\Pr[d_i(U_n)=1] = \sum_j p_j \cdot \Pr[d_i(z_n^{(j)})=1]$ 。

8.15 证明如果 G 是定义 8.12 意义下的强伪随机数发生器, 则其在定义 8.1 意义下也是一个伪随机数发生器。

提示: 考虑一个随机数序列, 使式 (8.2) 中的概率取最大值。

8.16 (**强计算不可区分性**) 提出定义 8.12 意义下计算不可区分性的定义 (即关于 (非一致的) 多项式规模电路的不可区分性)。证明以下两个断言。

(1) 关于 (非一致的) 多项式规模电路的计算不可区分性远远强于定义 8.4。

(2) 关于 (非一致的) 多项式规模电路的计算不可区分性在 (多项式数量) 多个样本下是不变的, 即使当概率空间不能在多项式时间内构造时也保持不变。

提示: 对第 1 部分, 见习题 8.10。对第 2 部分, 注意: 在命题 8.6 证明中生成的样本可以嵌入到区分电路中。

8.17 证明构造 8.17 在正则去随机化意义下可能会失效。具体地, 证明对于如定理 8.18 证明中提出的正则去随机发生器 G' , 构造 8.17 会失效。

8.18 结合定义 8.4 (并假设 $l(k) > k$), 证明存在规模为 $O(2^k \cdot l(k))$ 的电路, 使式 (8.11) 不成立。

提示: 电路可集成 G 定义域中所有值, 并将其输入与这些值相比较而进行判定。

8.19 (**对于定理 8.18, 构造一个集合系统**) 对任意 $\gamma > 0$, 构造一个如定理 8.18 的条件 (2) 中的集合系统 S , 其中 $m(k) = \Omega(k)$, $l(k) = 2^{\Omega(k)}$ 。

提示: 不失一般性, 假设 $\gamma < 1$, 并令 $m(k) = (\gamma/2) \cdot k$ 及 $l(k) = 2^{\gamma m(k)/6}$ 。用迭代方法构造集合系统 $S_1, S_2, \dots, S_{l(k)}$, 选择 S_i 为 $[k]$ 的前 $m(k)$ 个子集, 与前面的集合 $S_1, S_2, \dots, S_{i-1}$ 的交集要足够小。这样一个集合 S_i 的存在性可以用概率方法 (见参考文献 [11]) 证明。具体地, 对于一个固定的 $m(k)$ -子集 S' , 一个随机的 $m(k)$ -子集与 S' 交集元素个数大于 $\gamma m(k)$ 的概率小于 $2^{-\gamma m(k)/6}$, 因为交集中平均元素个数为 $(\gamma/2) \cdot m(k)$ 。因此, 一个随机的 $m(k)$ -子集与前 $i-1 < l(k) = 2^{\gamma m(k)/6}$ 个子集中每一个相交于至多 $\gamma m(k)$ 个点的概率为正。注意: 可在时间 $\binom{k}{m(k)} \cdot (i-1) \cdot m(k) < 2^k \cdot l(k) \cdot k$ 内构造 S_i , 从而 S 可在时间 $k 2^k \cdot l(k)^2 < 2^{2k}$ 内计算。

8.20 (**伪随机性与不可预测性, 电路解释**) 续习题 8.12, 证明如果存在一个规模为 s 的电路, 可以以缝隙 δ 区分 Z_n 与 U_l , 则存在 $i < l = |Z_n|$ 及规模为 $s + O(1)$ 的电路, 给定 Z_n 的前 i 比特时, 可以以至少 $\frac{1}{2} + \frac{\delta}{l}$ 的概率成功猜测第 $i+1$ 比特。

提示: 定义混合如定义 8.12, 注意: 对某个 i , 给定电路能以至少 δ/l 的缝隙区分第 i 个混合与第 $i+1$ 个混合。

8.21 假设构造 8.17 中的集合 S_i 都不相交, 且 $f: \{0,1\}^m \rightarrow \{0,1\}$ 是 T -不可近似的。证明对任意规模为 $T - O(1)$ 的电路 C , 有 $|\Pr[C(G(U_k)) = 1] - \Pr[C(U_l) = 1]| < 1/T$ 。

提示: 利用习题 8.20 证明逆否命题成立。注意: $G(U_k)$ 的第 $i+1$ 比特与其前 i 个比特统计无关, 从而, 如果能对其进行预测, 则得到了对 f 的近似。事实上, 可以利用平均法固定 $G(U_k)$ 的前 i 个比特来得到这样一个近似算法。

8.22 (**推广的定理 8.18**) 令 $l, m, m', T: \mathbb{N} \rightarrow \mathbb{N}$ 满足 $l(k)^2 + \tilde{O}(l(k) 2^{m'(k)}) < T(m(k))$ 。假

设以下两个条件成立。

(1) 存在一个指数时间计算的函数 $f: \{0,1\}^* \rightarrow \{0,1\}$ 是 T -不可近似的。

(2) 存在一个指数时间计算的函数 $S: \mathbb{N} \times \mathbb{N} \rightarrow 2^{\mathbb{N}}$, 使得对任意 k 及 $i=1,2,\dots,l(k)$, 有 $S(k,i) \subseteq [k]$ 及 $|S(k,i)| = m(k)$, 且对任意 k 及 $i \neq j$, 有 $|S(k,i) \cap S(k,j)| \leq m'(k)$ 。

证明利用构造 8.17 中定义的 G , 并令 $S_i = S(k,i)$, 可以得到一个扩张为 l 的正则去随机发生器。

提示: 根据定理 8.18 的证明, 只需注意用于对 $f(U_{m(k)})$ 求近似的电路具有规模 $l(k)^2 + l(k) \cdot \tilde{O}(2^{m'(k)})$, 且成功概率至少是 $(1/2) + (1/7l(k))$ 。

8.23 (定理 8.19 的第 2 部分) 证明如果对任意多项式 T , ε 中存在一个 T -不可近似的谓词, 则 $BPP \subseteq \bigcap_{\varepsilon>0} D_{\text{TIME}}(t_\varepsilon)$, 其中 $t_\varepsilon(n) \stackrel{\text{def}}{=} 2^{n^\varepsilon}$ 。

提示: 利用命题 8.15, 该命题充分表明, 对任意多项式 p 及任意常数 $\varepsilon > 0$, 存在一个扩张为 $l(k) = p(k^{1/\varepsilon})$ 的正则去随机发生器。该发生器可表示为习题 8.22 的形式, 并令 $m(k) = \sqrt{k}$, $m'(k) = O(\log k)$, 以及 $T(m(k)) = l(k)^2 + \tilde{O}(l(k)2^{m'(k)})$ 。注意: T 是一个多项式, 再利用习题 8.19, 可得到一个满足习题 8.22 要求的集合系统 (对上述参数), 并利用定理 7.10 可证。

8.24 (正则去随机化蕴含着困难问题) 证明定理 8.19 中每一部分的困难性假设都是相应正则去随机发生器存在的必要条件。更一般地, 证明扩张为 l 的正则去随机发生器的存在性蕴含着 ε 中存在 T -不可近似的谓词, 其中 $T(n) = l(n)^{1/O(1)}$ 。

提示: 主要说明如何得到 ε 中可用规模为 l 的电路计算的谓词, 并注意应用 7.2.1.3 节中的技术, 可以证明上述断言。给定一个正则去随机发生器 $G: \{0,1\}^k \rightarrow \{0,1\}^{l(k)}$, 考虑谓词 $f: \{0,1\}^{k+1} \rightarrow \{0,1\}$, 满足 $f(x) = 1$ 当且仅当存在 $s \in \{0,1\}^{l(k)}$, 使得 x 为 $G(s)$ 的前缀。注意: f 属于 ε , 且由计算 f 的算法可以得到 $G(U_k)$ 与 $U_{l(k)}$ 的区分器。

8.25 (对 (s, ε) -伪随机数发生器扩张的限制) 针对定义 8.20, 证明以下对于 (s, ε) -伪随机数发生器的扩张 l 的上限。

(1) 如果 $s(k) \geq 2$ 且 $\varepsilon(k) \leq 1/2$, 则 $l(k) < \varepsilon(k) \cdot (k+2) \cdot 2^{k+2-s(k)}$ 。

(2) 对任意 $s(k) \geq 1$ 及 $\varepsilon(k) < 1$, 有 $l(k) < 2^k$ 。

提示: 结合习题 8.37 与习题 8.38, 可证明第 2 部分。对第 1 部分, 用反证法, 考虑扩张为 $l(k) = \varepsilon(k) \cdot (k+2) \cdot 2^{k+2-s(k)}$ 的发生器及该发生器的所有 2^k 个输出 (关于 k 比特长种子): $\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(2^k)} \in \{0,1\}^{l(k)}$ 。构造一个非一致的空间为 s 的自动机, 如果对某个 $i \in [l(k)/(k+2)]$, 有 $x_{(i-1)(k+2)+1} \dots x_{i(k+2)}$ 等于 S_i 中某个串, 则该自动机接受 $x_1, x_2, \dots, x_{l(k)} \in \{0,1\}^{l(k)}$, 其中 S_i 中包含串 $\alpha^{((i-1)2^{s(k)-1}+1)}, \dots, \alpha^{(i2^{s(k)-1})}$ 到系数 $(i-1) \cdot (k+2) + 1, \dots, i \cdot (k+2)$ 上的投影。注意: 这样一个自动机至少接受发生器的 $(l(k)/(k+2)) \cdot 2^{s(k)-1} = 2\varepsilon(k) \cdot 2^k$ 个可能输出, 而一个随机 ($l(k)$ 比特) 串被接受的概率至多为 $(l(k)/(k+2)) \cdot 2^{(s(k)-1)-(k+2)} = \varepsilon(k)/2$ 。

8.26 ((s, ε) -伪随机数发生器的存在性) 与习题 8.25 相反, 对任意满足

$$s(k) < k - 2\log_2(k/\varepsilon(k)) - O(1)$$

的 s 及 ε , 证明存在扩张为 $l(k) = \Omega(\varepsilon(k)^2 \cdot 2^{k-s(k)} / s(k))$ 的 (s, ε) -伪随机数发生器 (不一定是高效的)。

提示: 利用习题 8.1 中的概率方法。注意: 非一致的空间为 s 时间为 l 的自动机可用长为 $l \cdot 2s2^s$ 的串来描述。

8.27 (多样本及空间受限的区分器) 假设两个概率空间 $\{X_k\}_{k \in \mathbb{N}}$ 和 $\{Y_k\}_{k \in \mathbb{N}}$ 由非一致自动机 (s, ε) -不可区分 (即任意空间为 s 的非一致的自动机的区分缝隙限制为函数 ε)。对任意函数 $t: \mathbb{N} \rightarrow \mathbb{N}$, 证明概率空间 $\left\{ \left(X_k^{(1)}, X_k^{(2)}, \dots, X_k^{(t(k))} \right) \right\}_{k \in \mathbb{N}}$ 和 $\left\{ \left(Y_k^{(1)}, Y_k^{(2)}, \dots, Y_k^{(t(k))} \right) \right\}_{k \in \mathbb{N}}$ 是 $(s, t\varepsilon)$ -不可区分的, 其中 $X_k^{(1)}$ 到 $X_k^{(t(k))}$ 及 $Y_k^{(1)}$ 到 $Y_k^{(t(k))}$ 是独立的随机变量, 每个 $X_k^{(i)}$ 与 X_k 相同, 每个 $Y_k^{(i)}$ 与 Y_k 相同。

提示: 利用混合技术。在区分第 i 个和第 $i+1$ 个混合时, 注意: 前 i 个块 (即 X_k 的副本) 以及后 $t(k) - (i+1)$ 个块 (即 Y_k 的副本) 可以固定并嵌入到非一致区分器之中。

8.28 更清晰地描述定理 8.21 证明中的发生器。

提示: 对于 $r \in \{0, 1\}^n$ 及 $h^{(1)}, h^{(2)}, \dots, h^{(t)} \in H_n$, 发生器输出由 2^t 个 n 比特串构成的序列, 满足第 i 个串等于 $h'(r)$, 其中 h' 是某些 $h^{(j)}$ 的组合。

8.29 (适应性 t -元独立测试) 称一个发生器 $G: \{0, 1\}^k \rightarrow \{0, 1\}^{l'(k)-b(k)}$ 是 t -元独立的, 如果对任意 t 个固定的块位置, 将分布 $G(U_k)$ 限制于这 t 个块上时在 $\{0, 1\}^{t \cdot b(k)}$ 上为均匀分布。证明通过测试 t 个块, 无法 (完美地) 区分一个 t -元独立的发生器的输出与均匀分布, 即使当被测试的块是自适应选择的 (即第 i 个块的位置基于前面已测试的 $i-1$ 个块的内容而选择)。

提示: 首先证明, 不失一般性, 只需考虑 (自适应地) 确定的测试者即可。然后证明该测试者可以看到 (在发生器的输出中) 适应性选择的位置上 t 个值构成的序列的概率等于 $2^{-t \cdot b(k)}$, 其中 $b(k)$ 为块长度。

8.30 (t -元独立发生器) 证明如命题 8.24 中定义的 G 可以产生 $\text{GF}(2^{b(k)})$ 上一个 t -元独立的序列。

提示: 对任意 t 个固定的下标 $i_1, i_2, \dots, i_t \in [l'(k)]$, 考虑 $G(U_k)_{i_1, i_2, \dots, i_t}$ 的分布 (即 $G(U_k)$ 在位置 $i_1, i_2, \dots, i_t$ 上的投影)。证明对于由 t 个可能值 $v_1, v_2, \dots, v_t \in \text{GF}(2^{b(k)})$ 的构成的任意序列, 存在唯一一个种子 $s \in \{0, 1\}^k$, 使得 $G(s)_{i_1, i_2, \dots, i_t} = (v_1, v_2, \dots, v_t)$ 。

8.31 (两两独立发生器) 作为热身, 考虑一种类似于命题 8.25 的构造, 不同之处在于这里的种子规定了一个 $b(k) \times m(k)$ 的仿射变换。也就是说, 对于 $s \in \{0, 1\}^{b(k) \cdot m(k)}$ 及 $r \in \{0, 1\}^{b(k)}$, 其中 $k = b(k) \cdot m(k) + b(k)$, 令

$$G(s, r) \stackrel{\text{def}}{=} (A_s v_1 + r, A_s v_2 + r, \dots, A_s v_{l'(k)} + r) \quad (8.23)$$

其中 A_s 是由串 s 确定的 $b(k) \times m(k)$ 矩阵。证明式 (8.23) 中的 G 是一个块长度为 b , 扩张为 l 的两两独立发生器 (注意: 在定理 7.7 的证明中有一个相关构造, 还可见习题 7.5)。然后,

证明式 (8.17) 中的 G 是一个块长度为 b , 扩张为 l 的两两独立发生器。

提示: 以下描述可同时应用于两种构造。首先注意到对任意给定的 $i \in [l'(k)]$, 序列 $G(U_k)$ 中的第 i 个元素, 记作 $G(U_k)_i$, 在 $\{0,1\}^{b(k)}$ 上是均匀分布的。实际上, 要证明对任意给定的 $s \in \{0,1\}^{k-b(k)}$, $G(s, U_{b(k)})$ 在 $\{0,1\}^{b(k)}$ 上均匀分布。下一步, 注意到只要证明对任意 $j \neq i$, 在已知 $G(U_k)_i$ 的值时, $G(U_k)_j$ 的值在 $\{0,1\}^{b(k)}$ 上均匀分布。这里关键的技术细节是要证明, 对任意非零向量 $v \in \{0,1\}^{m(k)}$ 及均匀选择的 $s \in \{0,1\}^{k-b(k)}$, 有 $A_s(v)$ (或 $T_s(v)$) 在 $\{0,1\}^{b(k)}$ 上均匀分布。对于随机的 $b(k) \times m(k)$ 矩阵, 要证明这一点是容易的, 对于随机的 Toeplitz 矩阵, 也可证明这个断言成立。

8.32 (适应性的 t -元独立测试) 注意: 与习题 8.29 相反, 关于非完善的不可区分性, 在检查 t 个位置的适应性与非适应性测试之间存在矛盾。

(1) 提出 2^{t-1} 比特的一种分布, 其中任意 t 个固定的比特位置诱导出一个分布, 与均匀分布是 $t \cdot 2^{-t}$ -接近的, 但是存在一种测试, 可以适应性地检查 t 个位置, 并以缝隙 $1/2$ 区分这种分布与均匀分布。

提示: 对 $((t-1) + 2^{t-1})$ 比特串的均匀分布进行修改, 使得前 $t-1$ 个位置指示着 (在剩余位置中) 一个被设为 0 的位置。

(2) 另一方面, 证明如果分布 X 中任意 t 个固定的比特位置诱导出一个与均匀分布为 ε -接近的分布, 则任意适应性地检查 t 个位置的测试能以缝隙 $2^t \cdot \varepsilon$ 区分 X 与均匀分布。

提示: 见习题 8.29。

8.33 假设 G 是一个扩张为 l 的 ε -偏移发生器。证明可以概率地检查 $l(k)$ -比特串 x 与 y 是否相等 (错误概率为 $(1+\varepsilon)/2$), 方法是比较 x 与 $G(s)$ 的模 2 内积和 y 与 $G(s)$ 的模 2 内积, 其中 $s \in \{0,1\}^k$ 为均匀选择。注意: 这种方法是对比较 x 与 r 的模 2 内积和 y 与 r 的模 2 内积的一个高效随机性近似, 其中 $r \in \{0,1\}^{l(k)}$ 为均匀选择。

提示: 考虑 $y = 0^{l(k)}$ 的特殊情况。

8.34 (偏移和与均匀分布的统计差) 令 X 为取值在 $\{0,1\}^t$ 上的随机变量。证明如果 X 与任意非空集的偏移至多为 ε , 则 X 与 U_t 的统计差不超过 $2^{t/2} \cdot \varepsilon$, 且对任意 $x \in \{0,1\}^t$, 有 $\Pr[X=x] = 2^{-t} \pm \varepsilon$ 。

提示: 考虑概率函数 $p: \{0,1\}^t \rightarrow [0,1]$, 定义为 $p(x) \stackrel{\text{def}}{=} \Pr[X=x]$, 并令 $\delta(x) \stackrel{\text{def}}{=} p(x) - 2^{-t}$ 表示 p 与均匀概率函数的偏差。将 $\{0,1\}^t$ 上的实值函数集看作是 2^t -维向量空间, 考虑该空间的两组正交基。其中第一个由 (Kroniker) 函数 $\{k_\alpha\}_{\alpha \in \{0,1\}^t}$ 组成, 满足当 $x = \alpha$ 时 $k_\alpha(x) = 1$, 否则, $k_\alpha(x) = 0$ 。第二个基由 (标准 Fourier) 函数 $\{f_S\}_{S \subseteq [t]}$ 组成, 定义为 $f_S(x_1 x_2 \cdots x_t) \stackrel{\text{def}}{=} 2^{-t/2} \prod_{i \in S} (-1)^{x_i}$ (其中 $f_\emptyset \equiv 2^{-t/2}$)^①。注意: X 与任意 $S \neq \emptyset$ 的偏差等于

① 验证两个基实际上是正交的 (即对任意 $\alpha \neq \beta$, 有 $\sum_x k_\alpha(x) k_\beta(x) = 0$, 且对任意 $S \neq T$, 有 $\sum_x f_S(x) f_T(x) = 0$), 并且是标准的 (即 $\sum_x k_\alpha(x)^2 = 1$ 且 $\sum_x f_S(x)^2 = 1$)。

$|\sum_x p(x) \cdot 2^{t/2} f_S(x)|$, 这又等于 $2^{t/2} |\sum_x \delta(x) f_S(x)|$ 。因此, 对任意 S (包括空集), 有 $|\sum_x \delta(x) f_S(x)| \leq 2^{-t/2} \varepsilon$, 这意味着 δ 的标准 Fourier 基表示中, 每个系数的绝对值不超过 $2^{-t/2} \varepsilon$ 。该向量系数的 2 阶范数上界为 $\sqrt{2^t \cdot (2^{-t/2} \varepsilon)^2} = \varepsilon$, 注意: 这里 δ 的范数是在 Kroniker 基之下, 从而两个断言得证。特别地, 2 阶范数在正交基下仍保持不变, 而最大范数的上界为 2 阶范数, 而 1 阶范数的上界是 2 阶范数的 $\sqrt{2^t}$ 倍。

8.35 ((非明确的)小偏移发生器的存在性) 证明对于 $k = \log_2(l(k)/\varepsilon(k)^2) + O(1)$, 存在函数 $G: \{0,1\}^k \rightarrow \{0,1\}^{l(k)}$, 满足 $G(U_k)$ 在任意非空子集 $[l(k)]$ 上的偏移量至多为 $\varepsilon(k)$ 。

提示: 利用习题 8.1 中的概率方法。

8.36 (小偏移 LFSR 发生器 (根据参考文献[10])) 利用提示 (并令 $t = k/2$), 分析以下根据定理 8.26 的构造 (图 8.5)。

(1) 证明 r_i 等于 $\sum_{j=0}^{t-1} c_j^{(f,i)} \cdot s_j$, 其中 $c_j^{(f,i)}$ 是 z^j 在 $(t-1)$ 次多项式中的系数, 该多项式是通过将 z^i 模多项式 $f(z)$ (即 $z^i \equiv \sum_{j=0}^{t-1} c_j^{(f,i)} z^j \pmod{f(z)}$) 而得到的。

提示: 由于 $z^t \equiv \sum_{j=0}^{t-1} f_j z^j \pmod{f(z)}$, 从而对任意 $i \geq t$, 有 $z^i \equiv \sum_{j=0}^{t-1} f_j z^{i-t+j} \pmod{f(z)}$ 。注意其与 $r_i = \sum_{j=0}^{t-1} f_j \cdot r_{i-t+j}$ 的联系。

(2) 对任意非空集合 $S \subseteq \{0, 1, \dots, l(k)-1\}$, 求序列 $r_0, r_1, \dots, r_{l(k)-1}$ 在 S 上的偏移量, 其中 f 为随机的 t 次不可约多项式, 且 $s = (s_0, s_1, \dots, s_{t-1}) \in \{0, 1\}^t$ 是均匀分布的。

① 对一个给定的 f 及随机的 $s \in \{0, 1\}^t$, 证明 $\sum_{i \in S} r_i$ 具有非零偏移量当且仅当 $f(z)$ 整除 $\sum_{i \in S} z^i$ 。

提示: 注意到 $\sum_{i \in S} r_i = \sum_{j=0}^{t-1} \sum_{i \in S} c_j^{(f,i)} s_j$, 并利用第 1 项。

② 证明一个随机 t 次不可约多项式整除 $\sum_{i \in S} z^i$ 的概率是 $\Theta(l(k)/2^t)$ 。

提示: 一个 n 次多项式最多可被 n/d 个不同的 t 次不可约多项式整除。另一方面, $\text{GF}(2)$ 上 d 次不可约多项式的个数为 $\Theta(2^d/d)$ 。

最后得到结论, 对随机的 f 和 s , 序列 $r_0, r_1, \dots, r_{l(k)-1}$ 的偏移量为 $O(l(k)/2^t)$ 。

注意: 在 LFSR 发生器的实现中, 需要将随机的 $k/2$ 比特串映射为几乎随机的 $k/2$ 次不可约多项式。这种映射可在 $\exp(k)$ -时间内构造, 如果 $l(k) = \exp(\Omega(k))$, 则 $\exp(k) = \text{poly}(l(k))$ 。参考文献[10]中第 8 节描述了一种利用 $O(k)$ -比特种子的更高效的映射。

8.37 (小偏移发生器的限制) 设 G 是扩张为 l 的 ε -偏移发生器, 并将 G 看作是从 $\text{GF}(2)^k$ 到 $\text{GF}(2)^{l(k)}$ 的映射。同样, G 输出中的每一个比特可以看作是关于 k 个输入变量 (每个在 $\text{GF}(2)$ 上取值) 的多项式^①。证明如果 $\varepsilon(k) < 1$, 且每个多项式的次数至多为 d , 则 $l(k) \leq \sum_{i=1}^d \binom{k}{i}$ 。

① 回顾 $\text{GF}(p)$ 上每个布尔函数可以表示为单变量次数至多为 $p-1$ 的多项式。

证明如下推论。

(1) 如果 $\varepsilon(k) < 1$ 则 $l(k) < 2^k$ (无论 d 是多少)。①

(2) 如果 $\varepsilon(k) < 1$ 则 $l(k) > k$, 则 G 不可能是一种线性变换。②

提示(对主要部分): 注意: 不失一般性, 上述所有多项式都有一个取值为 0 的自由项(且每个变量的次数至多为 1)。然后, 考虑由所有 k 个变量的 d -单项式张成的向量空间(即至多含 d 个变量的单项式)。由于 $\varepsilon(k) < 1$, 代表 G 输出比特的多项式必须对应于该空间中的独立向量序列。

8.38 (空间受限伪随机性的合理性检查) 我们提出以下事实作为针对空间受限自动机的伪随机数发生器的合理性检查。这个事实(其证明作为练习)是, 对任意 $\varepsilon(\cdot)$ 和 $s(\cdot)$, 满足对任意 k , 有 $s(k) \geq 1$, 如果 G 是 (s, ε) -伪随机的(如定义 8.20), 则 G 是一个 ε 偏移发生器。

8.39 与习题 8.38 相反, 证明存在 $\{0, 1\}^n$ 上 $\exp(-\Omega(n))$ -偏移的分布, 不是 $(2, 0.666)$ -伪随机的。

提示: 证明集合

$$\left\{ \sigma_1 \sigma_2 \cdots \sigma_n : \sum_{i=1}^n \sigma_i \equiv 0 \pmod{3} \right\}$$

上的均匀分布偏移量为 $\exp(-\Omega(n))$ 。

8.40 (t -元独立发生器的近似(根据参考文献[170])) 将定理 8.26 中的小偏移发生器与式 (8.16) 中的 t -元独立发生器相结合, 并根据后者的线性性, 构造一个可以生成 l -比特长序列的发生器, 其中任意 t 个位置与均匀分布的距离至多为 ε (指变异距离), 且种子长度为 $O(t + \log(1/\varepsilon) + \log \log l)$ (对于最大范数, 种子长为 $O(\log(t/\varepsilon) + \log \log l)$ 即可)。

提示: 首先要注意, 对任意的 t, l' 及 $b \geq \log_2 l'$, 式 (8.16) 上的变换可以用一个固定的 (GF(2) 上的) 线性变换来实现, 该变换将 $t \cdot b$ -比特种子扩展为 l 比特随机序列, 其中 $l = l' \cdot b$ 。从而, 对于 $b = \log_2 l'$, 存在一个固定的 GF(2) 上线性变换 T , 将一个长为 $t \cdot b$ 的随机种子扩展为 t -元独立的长为 l 的比特序列(即 $TU_{t,b}$ 在 $\{0, 1\}^l$ 上是 t -元独立的)。因此, T 的任意 t 行线性无关。关键在于将上述随机种子用一个 ε' -偏移的序列代替时, 输出序列中任意 $i \leq t$ 个位置都可诱导出偏移量至多为 ε' 的分布(因为这些比特对于用作种子的 ε' -偏移序列定义了一个非零线性测试)。注意: 新种子(用于生成长为 $t \cdot b$ 的 ε' -偏移序列)的长度为 $O(\log(t/\varepsilon'))$ 。利用习题 8.34 可得到结论, 任意 t 个位置与均匀分布的距离最多为 $2^{t/2} \cdot \varepsilon'$ (变异距离)。利用长为 $O(\log(t/\varepsilon')) + \log \log l$ 的种子, 可以得到这个结论, 再令 $\varepsilon' = 2^{-t/2} \cdot \varepsilon$, 断言得证。

8.41 (小偏移发生器与纠错码) 证明扩张为 l 的 ε -偏移发生器与二元线性纠错码(见附录 E.1.1)之间存在对应关系, 这种码将 $l(k)$ -比特串映射为 2^k 比特串, 使得任意两个码字

① 这个上界是最优的, 因为(高效的)扩张为 $l(k) = \text{poly}(\varepsilon(k)) \cdot 2^k$ 的 ε -偏移发生器确实存在(见参考文献[170])。

② 相反, 双线性的 ε -偏移发生器(即 $l(k) > k$) 确实存在, 例如, $G(s) = (s, b(s))$, 其中 $b(s_1, s_2, \dots, s_k) = \sum_{i=1}^{k/2} s_i s_{(k/2)+i} \pmod{2}$, 是一个 ε -偏移发生器, 这里 $\varepsilon(k) = \exp(-\Omega(k))$ 。

提示: 主要考虑包含最后一个输出比特的集合上的偏移量, 不失一般性, 证明这足以用于分析 $b(U_k)$ 的偏移量。

的距离为 $(1 \pm \varepsilon(k)) \cdot 2^{k-1}$ 。

提示: 将 $\{0,1\}^k$ 与 $[2^k]$ 相关联, 则发生器 $G: [2^k] \rightarrow \{0,1\}^{l(k)}$ 对应于码 $C: \{0,1\}^{l(k)} \rightarrow \{0,1\}^{2^k}$, 使得对任意 $i \in [l(k)]$ 及 $j \in [2^k]$, $G(j)$ 的第 i 比特等于 $C(0^{i-1}10^{l(k)-i})$ 的第 j 比特。

8.42 (有限域上序列的偏移量) 对于素数 p , 设 ζ 为 $\text{GF}(p)$ 上的随机变量, 且 $\delta(v) \stackrel{\text{def}}{=} \Pr[\zeta = v] - (1/p)$ 。证明 $\max_{v \in \text{GF}(p)} \{|\delta(v)|\}$ 的上界为 $b = \max_{c \in \{1, 2, \dots, p-1\}} \{ \|E[\omega^{c\zeta}]\| \}$, 其中 ω 表示 p 次本原单位 (复) 根, 且 $\sum_{v \in \text{GF}(p)} |\delta(v)|$ 的上界为 $\sqrt{p} \cdot b$ 。

提示: 类似于习题 8.34, 将 $\text{GF}(p)$ 上的概率分布看作是 p -维向量, 且考虑 $\text{GF}(p)$ 上的复函数集合的两组基: Kroniker 基 (即 $x=i$ 时 $k_i(x)=1$, 否则, $k_i(x)=0$) 及 (正规) Fourier 基 (即 $f_i(x) = p^{-1/2} \cdot \omega^{ix}$)。注意: ζ 的偏移量对应于 δ 与非常数 Fourier 函数的内积, 而 ζ 与均匀分布的距离对应于 δ 与 Kroniker 函数的内积。

8.43 (扩张随机漫游定理的另一个版本) 利用定理 8.28 中的符号, 证明长为 l' 的随机漫游停留在 W 中的概率至多为 $(\rho + (\lambda/d)^2)^{l'/2}$ 。事实上, 可以证明一个更具一般性的断言, 针对着长为 l' 的随机漫游与 $W_0 \times W_1 \times \dots \times W_{l'-1}$ 相交的概率。上限为

$$\sqrt{\rho_0} \cdot \prod_{i=1}^{l'-1} \sqrt{\rho_i + (\lambda/d)^2} \quad (8.24)$$

其中

$$\rho_i \stackrel{\text{def}}{=} |W_i|/|V|。$$

提示: 将随机漫游看作是相应的概率向量在适应变换下的演化。这种变换对应于在图上随机前进一步, 并通过一个“筛”, 其中只保留对应于当前集合 W_i 的项。注意: 第一个变换对与均匀分布 (是扩张图的邻接矩阵的第一个特征) 正交的部分进行了压缩, 而第二个变换对在均匀分布方向上的部分进行压缩。更多细节见附录 E.2.1.3。

8.44 利用定理 8.28 的符号, 证明长为 l' 的随机漫游经过 W 的次数大于 $\alpha l'$ 的概率小于 $\binom{l'}{\alpha l'} \cdot (\rho + (\lambda/d)^2)^{\alpha l'/2}$ 。例如, 当 $\alpha = 1/2$ 且 $\lambda/d < \sqrt{\rho}$ 时, 得到的上界为 $(32\rho)^{l'/4}$, 我们指出, 可以得到更好的上界 (如可参考文献[120])。

提示: 利用对所有 $m = \alpha l'$ 次访问的序列的联合上限, 并利用式 (8.24) 求在第 $j_1, j_2, \dots, j_m$ 步访问 W 的概率上界, 其中当 $i \in \{j_1, j_2, \dots, j_m\}$ 时, $W_i = W$, 否则, $W_i = V$ 。

第9章

概率证明系统

证明才能让我相信。

Shimon Even (1935—2004)

发现证明是一件富于创造性的工作，它带来的荣誉感使人们忽视了证明的验证这个较少被赞美的过程，然而，后者才能真正赋予证明价值。在概念意义上，证明要排在验证之后，而从技术上看，证明系统甚至是用其验证程序定义的。

验证程序的定义中使用计算的概念，特别是高效的计算。 NP 类的定义使其更加明确，其中将高效计算与确定的多项式时间算法相关联。然而，在以下讨论中，如果不拘泥于传统，允许概率的验证程序，则将有更多的收获。

本章研究3种类型的概率证明系统，分别称为交互式证明、零知识证明和概率可检验证明。针对每一类都给出很重要的结论，在仅考虑确定性证明时，是无法得出这些结论的。

内容提要

将高效计算与确定型多项式时间程序相关联，这是将 NP -证明系统视为规范证明系统（具有高效的验证程序）的基础。允许概率型验证程序并利用统计方法，将得到各种类型的概率证明系统。这些证明系统以一定概率产生误差（上限明确且可通过多次应用来减少误差），然而，与传统的证明系统（确定型的，无误差）相比，它们有各种优越性。

随机性和交互式验证程序促成交互式证明系统的诞生，这类证明系统比确定的证明更加强大。特别地， $PSPACE \supseteq coNP$ 中的任意集合都存在交互式证明系统（如在不可满足的谓词公式集合中），而普遍认为 $coNP$ 中某些集合没有 NP -证明系统（即 $NP \neq coNP$ ）。我们强调这里的“证明”并非固定和静态的对象，而更像是一种随机（和动态的）过程，其中验证者与证明者进行交互。直观上，人们会认为交互是指由验证者提出问题，而证明者必须作出令人信服的回答。

随机性和交互式验证程序还涉及到有意义的零知识证明概念，它在理论和应用上都有重要价值（特别是在密码学中）。不严格地讲，在零知识证明中，（验证者）除了确信断言为真之外，得不到任何知识。例如，对某个谓词公式可满足性的零知识证明并不会揭示一种可满足的赋值，或这种赋值的任何一部分信息（如第一个变量是否取值为真）。因此，对零知识证明的成功验证在确认证言的有效性上与得不到任何其他信息（在收到这样的确认证明之后）之间形成极端的对比。人们发现，在合理的复杂性假设下（即假设单向函数存在）， NP 中任意集合都存在零知识证明系统。

NP-证明可以高效地转化为另一种（冗余）形式，以在证明中要检查的（随机）位置数量与和正确性确认之间取得折中。特别地， NP 中任意集合具有支持概率验证的 NP-证明系统，使得误差概率随着证明读取的比特数量而呈指数降低。这种冗余形式的 NP-证明称为概率可检验证明（或 PCP）。除了在概念上有意义之外，PCP 与大量近似问题的复杂性研究密切相关。

引言及预备知识

在概念意义上，证明应排在验证之后。事实上，在数学和现实生活中，证明只在被普遍接受的推理原则下才有意义，而验证过程实际上是检查证明过程是否正确使用了这些原理。因此，一些通常被认为是理所当然的原则，比使用它们的证明更重要。对任何事物进行推理，都必须基于被普遍接受的推理原则之上。

推理原则与验证程序相关，验证程序能辨别是否正确使用了这些原则。一个推理过程关于某条给定原则是合理的（从而被视为证明）当且仅当推理中正确使用了这些基本原则。因此，一个推理过程被认为是合理的当且仅当它可被相应的验证程序接受。这意味着，从技术上讲，证明是用给定验证程序来定义的（或关于该验证程序定义的）。证明的形式化（即逻辑）研究中很好地阐明了这种情况，形式化证明实际上研究有严格定义的证明系统：这些证明系统是利验证程序术语定义的（通常是明确定义，有时也不太明确）。

验证程序的概念建立于计算之上。这解释了历史上逻辑学家们为什么会对计算机科学感兴趣（见参考文献[55, 225]）。另外，通常总认为程序的验证十分容易，从而自然地在验证程序与高效计算之间建立起联系。复杂性理论用 NP 和 NP-证明系统来解释这种联系（见定义 2.5），从而使其更加清晰，NP-证明系统的研究目标是所有高效的验证程序^①。

回顾在定义 2.5 中，用确定的多项式时间算法来识别高效（验证）程序，并明确地限制证明长度为断言长度的多项式。因此，验证程序的执行步骤数是断言长度的多项式。我们指出，确定型证明系统允许更长的证明（但是要求对于该证明的长度，验证是高效的），并用足够长的（对断言的）填充将证明模型化为 NP-证明系统。

事实上，如果将高效程序与确定型多项式时间算法相关联，则 NP-证明是可高效验证证明（即具有高效验证程序的证明系统）的终极定义。然而，我们将看到，采用非传统方式而允许概率（多项式时间）算法，特别是概率的验证程序时，将得到更丰富的结论。

- 随机及交互式的验证程序似乎比其确定版本功能更强大。
- 交互证明系统可用于构造（有意义的）零知识证明，后者具有重要的理论和实践意义。
- NP-证明可以高效地转化为一种支持超高速概率验证的（冗余）形式，转化时只需在证明中加入少量随机因素即可。

上述这些情形中都明确规定了验证程序的计算复杂性上限，验证程序通常又被拟人化为验证者。进一步地，所有证明系统都允许验证者生成随机数并遵守统计规则。因此，所有证明系统都会以一定的概率出错，然而，错误概率有明确界限，并且可通过多次证明来减少错误概率。

一个习惯

在介绍一个证明系统时，我们用待证明断言的长度（被看作是对验证者的输入）表示所

^① 相反，传统证明系统的定义建立于推理规则之上，在传统证明环境中这很正常。从推理规则得到高效验证程序，这仅仅是在高效计算过程间进行信息交互的结果。

有的复杂性上限。也就是说,“多项式时间”指时间为断言长度的多项式。这种用法在所有情形中都很自然。上述对 NP-证明系统的讨论也遵循这种习惯。^①

常用符号

我们用 poly 表示所有上限为多项式的整数函数,用 $\log$ 表示所有上限为对数函数(即 $f \in \log$ 当且仅当 $f(n) = O(\log n)$) 的整数函数。本章提到的所有复杂性测试都假设可在多项式时间内构造。

内容组织

9.1 节提出交互式证明系统的基本定义及相关结论。交互式证明系统的定义是 9.2 节介绍的零知识证明的基础。9.3 节提出概率可检验证明(PCP)的定义及相关结论,与其他两节是独立的。

预备知识

假设读者具备概率论(见附录 D.1)和随机化算法(见 6.1 节)的基础知识。

9.1 交互式证明系统

随着随机化和交互式计算日渐被人接受,人们很自然地将高效计算的概念与概率和交互式多项式时间计算相关联。由此得到交互式证明系统,其中验证程序是交互式和随机的。因此,这种情形下的“证明”也不是一个固定和静态的过程,而是随机(和动态的)过程,其中验证者与证明者进行交互。直观上可以认为交互中由验证者提出的问题,而证明者必须作出相应的令人信服的回答。

上述讨论及 9.1.1 节中的定义明确涉及到证明者,而在传统的证明系统定义中(即 NP-证明系统),证明者是模糊不清的。在给出定义之前,我们强调并进一步讨论这些概念问题。

9.1.1 动机和观点

我们将讨论文献中对于证明给出的各种不同解释,以及这些解释中体现的不同观点。要强调定义交互式证明的动机不是为了取代数证明,而是要理解其他形式的证明。具体而言,我们将对比“书面证明”与“交互式证明”,强调证明中“证明者”与“验证者”的角色,并讨论证明的完备性与合理性概念(一些读者可能会发现看过 9.1.2 节之后再回到本节是有大有裨益的)。

9.1.1.1 静态对象与交互式过程

在数学中有一个传统,认为“证明”就是一系列的断言,这些断言或者是不证自明的,或者是由不证自明的规则推导出来的。实际上,从概念和技术的角度看,用“被普遍接受”代替“不证自明”会更准确(不证自明其实就是被普遍接受)。事实上,在证明的形式化(即逻辑)研究中,被普遍接受的断言称为公理,而被普遍接受的规则称为推理规则。我们强调数学证明的一个关键性质:这些证明都是固定的(静态)对象。

相反,在人类的其他活动领域,“证明”这一概念有更广泛的含义。在许多情况下,证明并非固定对象,而是一个过程,通过它来确认一个断言的合理性。例如,在法律中,能经受

^① 回顾定义 2.5 涉及到对证明的多项式时间验证,而验证长度必须是断言长度的多项式。

对手的交互讯问（对手可能会提出苛刻的和/或欺骗性的问题），被看作是证人对其证据的证明。类似地，日常生活中的各种争论中也隐含着类似的对某些断言的证明。这在哲学和政治辩论中尤为常见，并且甚至出现在科学辩论中。当然，这些争论的一个关键性质是其（“动态的”）交互性。有趣的是，“交互性证明”的诱人之处体现在，它们被一些采用反证法的数学证明所模仿（以严格方式），实际上是模仿与潜在（真实的）怀疑者之间想象中的辩论。

数学证明与各种“日常证明”间的另一个区别是，前者追求确定性，而后者（“仅仅”）追求针对合理的质疑证明断言。有争议的是，有明确上限的错误概率（在交互式证明系统定义中所指的）是针对任意合理质疑证明断言的一种极强形式。

还要注意，数学中通常认为证明比结论（即定理）更重要。相反，在许多日常情况下，证明居于结论之后（在重要性上）。伴随着这些矛盾的态度，产生了书面证明和“交互式”证明间的差别：如果证明本身有意义，则必须将其存档；而如果只重视要证明的结论，则通过何种方式证明并不重要。

有趣的是，对本章而言，只要采取上述日常态度（而非数学态度）就足够了，其中将证明视为验证断言正确性的工具（这在零知识证明中达到极端，其中要求除了使人确信断言的正确性之外，证明本身毫无用处）。

一般情况下，人们喜欢对各种形式的证明建立模型，并主要考虑那些可以用自动程序验证的证明。这些验证程序用于检查证明的正确性，并且显然可验证人们要求的一些附加特征，如美丽、富于洞察力等。本节讨论能被高效验证的具有最一般形式的证明系统。

要注意，我们研究的证明系统涉及到一些寻常的定理（如断言某个谓词公式不可满足，或一方根据预定协议发送了信息）。这些关于证明系统的定理将以传统的数学方式得到证明。

9.1.1.2 证明者和验证者

一般人们理解的证明系统中包含交互式的验证过程，这要求引入两个交互的参与方，称为证明者和验证者。验证者运行验证程序，该程序包含于证明系统的定义中。证明者则尝试使验证者确信。在静态（或非交互式）证明中，证明者是提供证明的一个超然实体，因此，通常（在讨论对证明的验证时）根本不提到证明者。然而，即便是在讨论这种静态（非交互式）证明时，明确指出潜在的证明者也是有益的。

我们强调在任何一个证明系统中，都要对证明者采取“不信任态度”。如果验证者信任证明者，就不需要任何证明了。因此，在讨论证明系统时，要想象这样一个环境，其中验证者不信任证明者，从而对证明者说的任何话都表示怀疑。此时，证明者的目标是使验证者确信，而验证者要确保自己没有被他证明者欺骗。（见 9.1.1.3 节中的进一步讨论。）注意验证者要利用预定的验证程序来“可信地”保护自身利益；事实上，这种信任上的不对称是由于重点讨论验证程序造成的。一般来说，每一方肯定要保护自身利益，但通常不认为一方要保护另一方的利益（不相信证明者要保护验证者的利益，使其不被证明者欺骗）。

两个参与方之间另一种不对称是我们主要考虑验证任务的复杂性，而忽略了（作为第一个近似）证明任务的复杂性（只在 9.1.5.1 节中讨论）。注意到这种不对称性体现在 NP -证明系统的定义中，其中要求验证过程是高效的，而对 $\text{NP} \setminus \text{P}$ 中的集合找到充分的证明则是不可行的。因此，我们首先考虑在与任意证明者（可能拥有无限计算能力）交互时，能被高效验证的是什么。解决了这个问题之后，再考虑证明任务的复杂度（见 9.1.5.1 节）。

9.1.1.3 完备性和合理性

证明系统（即验证程序）的两个基本性质是合理性（或正确性）和完备性。合理性要求验证者不能被“欺骗”而接受错误断言。换言之，合理性体现了验证者保护自己不被错误断言欺骗的能力（无论证明者为达到欺骗的目的而做了什么）。另一方面，完备性则体现了某些证明者使验证者相信正确断言（属于某个指定的正确断言集合）的能力。注意：两个性质在证明系统的定义中都很重要。

并非所有正确断言集合都存在“合理的”证明系统，可以证明每一个断言成立（而不“证明”错误断言）。一些结论赋予这个基本现象准确的含义，包括哥德尔的不完备性定理（Gödel's Incompleteness Theorem）和图灵关于停机问题不可判定性的定理。相反，回顾 NP 类的定义，是指具有支持高效的确定性验证（或“书面证明”）的证明系统的问题类。本节研究高效验证程序一个更广泛的定义（既允许随机化也支持交互性）。

9.1.2 定义

大体上，可将交互式证明看作是计算受限的验证者与计算无限的证明者之间的“游戏”，其中证明者的目标是使验证者确信某些断言的正确性。具体而言，验证者使用一个概率多项式时间策略（而对证明者策略无计算上的限制）。如果断言成立，则验证者总是接受（即与适当的证明者策略交互时）。另一方面，如果断言错误，则无论证明者采取何种策略，验证者也必须以至少 $1/2$ 的概率拒绝该断言（可以通过多次运行证明系统来减少错误概率）。

我们利用参与方使用的策略来形式化定义交互过程。^①参与方策略是一个函数，将其在交互中得到的信息映射为下一步行动；即策略中描述（或规定）参与方的下一步行动（即要发出的下一个消息或最终判定）是公共输入（即上述断言）、参与方的内部随机数，以及目前为止收到信息的函数。注意：这一定义（不明确地）假设各个参与方记录下了其以往使用的所有随机数，以及接收到的所有信息，并基于这些信息确定下一步行动。因此，分别使用策略 A 和策略 B 的两方之间的一次交互，由公共输入（记作 x ）和双方的随机数（分别记作 r_A 和 r_B ）来确定。假设交互由 A 开始（且由 B 结束），则相应的（ t -轮）交互的记录（对于公共输入 x 和随机数 r_A 和 r_B ）为 $\alpha_1, \beta_1, \dots, \alpha_t, \beta_t$ ，其中 $\alpha_i = A(x, r_A, \beta_1, \beta_2, \dots, \beta_{i-1})$ $\beta_i = B(x, r_B, \alpha_1, \alpha_2, \dots, \alpha_i)$ 。相应地， A 的最终判定定义为 $A(x, r_A, \beta_1, \beta_2, \dots, \beta_t)$ 。

如果一个参与方在计算下一步行动时，所需步骤数是公共输入长度的多项式，则称其采用了概率多项式时间策略。特别地，这意味着对于公共输入 x ，该策略只考虑 $|x|$ 的多项式个消息，且每个消息的长度为 $\text{poly}(|x|)$ 。^②直观上，如果另一方发送的消息总长度超过上限（ $|x|$ 的多项式），则其执行被挂起。因此，根据上述策略，如果对任意 i 和 $r_A, \beta_1, \beta_2, \dots, \beta_i$ ， $A(x, r_A, \beta_1, \beta_2, \dots, \beta_i)$ 的值可在 $|x|$ 的多项式时间内求出，则称 A 为概率多项式时间策略。这里再次要求 $|r_A|$ 、 t 和 $|\beta_i|$ 均为 $|x|$ 的多项式。

定义 9.1（交互式证明系统——IP）^③ 集合 S 的交互式证明系统是在证明者和验证者之

① 另一种定义涉及到交互式机器，它能体现每一方的行为（如可见于[91, 4.2.1.1 节]）。这种机器调用相应策略，处理与另一方的通信，并记录收到的所有信息。

② 显然，多项式时间策略中的内部随机数数量上界必须为 x 长度的多项式。

③ 我们采用了对验证者和证明者策略均给出规定的习惯做法。另一种做法是只规定验证者策略，并将完备性条件重述如下：存在一种证明者策略 P ，使得对任意 $x \in S$ ，验证者 V 在关于公共输入 x 与 P 交互之后，总是接受 x 。

间进行的一个两方游戏，其中验证者执行概率多项式时间策略，记作 V ，而（计算能力无限的）证明者执行策略 P ，满足以下两个条件。

- 完备性。对任意 $x \in S$ ，验证者 V 关于公共输入 x 与证明者 P 交互之后，总是接受 x 。
- 合理性。对任意 $x \notin S$ 及策略 P^* ，验证者 V 关于公共输入 x 与 P^* 交互之后，以至少 $1/2$ 的概率拒绝 x 。

将具有交互式证明系统的集合类记作 IP 。

通过多次重复运行证明系统，可以减小（合理性条件中的）错误概率（在串行重复中这很显然，而在并行重复中也成立，见习题 9.1）。特别地，证明过程重复 k 次之后，验证者被欺骗（即接受一个错误断言）的概率将减少至 2^{-k} ，而对任意 $k = \text{poly}(|x|)$ ，重复 k 次都是可以做到的。9.1.4 节将讨论基本定义的变形。

随机性的作用

随机性在交互式证明中起着重要作用，如果限制验证者只能采取确定策略，得到的交互式证明系统在 NP-证明系统中将不具备任何优势。原因在于，如果验证者是确定的，则证明者可以预知验证者在交互中的行为。因此，证明者只需提供对验证者（预知的）提问序列的应答序列，而验证者检查这些结果，会发现它们都是可信的。实际上，合理性条件中的错误（不仅仅针对随机化验证）对于交互证明系统而言非常重要（即其超出 NP-证明的能力）。

命题 9.2 假设 S 具有交互证明系统 (P, V) ，其中无合理性错误，即对任意 $x \notin S$ 及任意策略 P^* ，验证者 V 关于公共输入 x ，与 P^* 交互之后，以概率 1 拒绝 x ，则 $S \in \mathcal{NP}$ 。

证明：不失一般性，可以假设 V 是确定型的（只须随意固定随机带内容（如令其为全 0 串），并注意此时（完善的）完备性和完善的合理性（即无错误）仍成立）。从而，合理性错误为零的情况可归结为确定型验证者。

现在由于 V 是确定的，证明者可以预测 V 发出的每一条消息（因为每个消息都由公共输入和证明者之前发出的信息唯一确定）。因此，仅仅根据公共输入 x ，就能（预先）确定（不与 V 交互）证明者的最优消息序列（使 V 接受 x 的消息序列）。^①从而 $x \in S$ 当且仅当存在（证明者）消息序列，使 V （确定地）接受 x ，而某个特定消息序列是否能令 V 接受 x 只取决于该序列及公共输入 x （因为 V 没有做出任何随机选择来影响这种判定）^②。上述条件是否成立可在多项式时间内检查，于是，所有的“通过序列”就构成了 $x \in S$ 的 NP-证据，从而 $S \in \mathcal{NP}$ 。 ■

思考

命题 9.2 证明中的推理体现了一个原则，即与一个行为易预测的参与方交互是毫无意义的，因为其行为在不交互时也能确定。这个原则代表了证明者的观点（关于与确定的验证者交互）。相反，即使是计算能力无限的参与方（如证明者）也可以从与行为不可预测的参与方（如随机的验证者）的交互中受益，因为通过交互可以得到有用信息（如关于验证者随机选择的信息，这又能使证明者做出令人信服的应答的概率增加）。进一步地，从验证者的角度看，与证明者交互也是有益的，因为后者有更强大的计算能力（从而即使证明者的行为从信息论角度是可预测的，验证者也不易预测其行为）。

① 照例我们并不关心确定该序列的复杂度，因为此时对证明者的计算能力没有任何限制。

② 当 V 为随机型时，其最终判定还依赖于内部生成的随机数（而不仅仅是公共输入和证明者的消息序列）。此时，由验证者自身的消息可能会揭示关于验证者内部随机数的信息，而这又能帮助证明者作出更令人信服的回答。

9.1.3 交互式证明的功能

我们已经了解到随机性在交互式证明系统中起着关键作用, 如果没有随机性, 则交互式证明并不比 NP-证明更强大。事实上, 交互式证明的威力来自于随机性与交互的组合。本小节首先用一个简单的证明系统阐述这一点, 该系统涉及到一类不具备 NP-证明系统的 coNP 集, 然后证明一个著名结论 $IP = PSPACE$, 表明交互式证明比 NP-证明功能更强大。

9.1.3.1 一个简单例子

一天, 在奥林匹斯山上, 目光炯炯的雅典娜宣称新的银盖罐子里的酒不如旧的饰金罐子里的酒甜。伟大的耶稣被这个说法激怒, 因为新罐子是赫拉让他用的。耶稣命人拿给雅典娜 100 个杯子, 每个杯子中随机地倒入旧罐或新罐中的酒, 让雅典娜尝每一杯酒, 并说出酒来自哪个罐子。令所有人惊奇的是, 聪明的雅典娜正确地识别出了每杯酒的来源, 于是, 众神之王说: “我的孩子, 你要么是正确的, 要么极其幸运。” 由于所有的神都知道幸运并非智慧女神雅典娜的属性, 所以他们都认为永远正确的女神这一次也是正确的。

上面的故事形象地描述了构造 9.3 中关于非同构图问题的交互式证明的主要思想。通俗地讲, 这个交互式证明系统可以证明两个给定对象是不同的 (在上面的故事中, 对象就是装酒的两个罐子, 而在构造 9.3 中, 是两个非同构的图)。注意: 一般来说, 证明对象间的相似性是容易的, 因为可以构造一个映射 (从一个对象到另一个对象) 来表现这种相似性。而要证明对象是不同的似乎不那么容易, 因为通常不存在证明对象不同的简便方法 (如证明一个特定映射不满足相似性显然是不够的, 必须枚举出所有的映射 (并证明每个映射都不满足相似性), 而枚举绝非一种简便方法)。更一般地, 对于给定对象, 要证明存在一种易验证的结构通常很容易, 只需提出这种结构即可, 但是证明不存在这样的结构则很困难。严格地说, NP 集的成员问题可通过提出 NP-证据来证明, 但是尚不清楚如何证明不存在这种 NP-证据。事实上, 通常认为 $\text{coNP} \neq \text{NP}$ 。

两个图 $G_1 = (V_1, E_1)$ 和 $G_2 = (V_2, E_2)$ 称为是同构的, 如果存在由顶点集 V_1 到顶点集 V_2 的 1-1 映射 φ , 使得 $\{u, v\} \in E_1$ 当且仅当 $\{\varphi(u), \varphi(v)\} \in E_2$ 。这种 (保持边的) 映射 φ 如果存在, 则称其为两个图之间的同构映射。以下协议给出一种证明两个图不同构的方法, 而是否存在证明此种结论的非交互式过程 (即通过一个 NP-证明系统) 尚且未知。

构造 9.3 (非同构图的交互式证明)

- 公共输入。两个图 $G_1 = (V_1, E_1)$ 和 $G_2 = (V_2, E_2)$ 。
- 验证者的第一步 (V_1)。验证者随机选择一个图, 并将该图的一个随机同构副本发送给证明者。就是说, 验证者随机选择 $\sigma \in \{1, 2\}$, 以及顶点集 V_σ 上的随机置换 π 。验证者构造一个图, 其顶点集为 V_σ , 边集为 $E = \overset{\text{def}}{\{\{\pi(u), \pi(v)\} : \{u, v\} \in E_\sigma\}}$, 并将 (V_σ, E) 发送给证明者。
- 动机说明。如果输入的图如证明者所称是不同构的, 则证明者应该能区分 (不一定用高效算法) 两个图的同构副本。然而, 如果输入的图是同构的, 则一个图的随机同构副本与另一个图的随机同构副本分布完全相同。
- 证明者步骤。在收到验证者传来的图 $G' = (V', E')$ 之后, 证明者求 $\tau \in \{1, 2\}$ 使得图 G' 与图 G_τ 同构 (如果 $\tau = 1, 2$ 均满足条件, 则任意选择 τ 即可; 当 $\tau = 1, 2$ 均不满足条件时, 令 $\tau = 0$)。证明者将 τ 传给验证者。

• 验证者的第二步 (V_2)。如果证明者传来的 τ 等于 σ (在步骤 V_1 中选择的), 则验证者输出 1 (即接受公共输入), 否则, 验证者输出 0 (即拒绝公共输入)。

构造 9.3 中的验证者策略易在概率多项式时间内实现。实现证明者策略的概率多项式时间算法尚且未知, 但也不要求。上述的动机说明表明, 构造 9.3 是非同构图问题的交互式证明系统。^①前面提到该问题属于 coNP -集 (并非 NP 的子集)。

9.1.3.2 交互式证明的丰富功能

构造 9.3 中的交互式证明系统针对一个具体的不属于 NP 的 coNP -集。人们发现交互式证明系统足以证明任何 coNP -集的成员问题 (如证明一个图不是 3-着色的)。因此, 假设 $\text{NP} \neq \text{coNP}$ 成立, 则交互式证明系统比 NP -证明系统功能更强大。进一步地, 具有交互式证明系统的集合类与可在多项式空间内判定的集合类完全重合。

定理 9.4 (IP 定理) $IP = PSPACE$ 。

普遍认为 NP 是 $PSPACE$ 的一个恰当子集, 因此, 在这个假设下, 交互式证明比 NP -证明更强大。

9.1.3.3 定理 9.4 证明概要

首先证明 $\text{coNP} \subseteq IP$, 方法是针对 coNP -完全的不可满足 CNF 范式集合, 构造一个交互式证明系统, 然后将该证明系统扩展到对 $PSPACE$ -完全的不可满足量化布尔公式集的证明。最后利用 $IP \subseteq PSPACE$ 。事实上, 证明某个 coNP -完全集具有交互式证明系统是证明定理 9.4 的核心 (见习题 9.2)。

我们用代数方法证明不可满足的 CNF 范式集合具有交互式证明系统, 代数方法被用于该布尔问题的算术推广 (而非问题自身) 中。也就是说, 为了证明一个布尔问题具有交互式证明系统, 首先引入 CNF 范式的算术推广, 再对得到的算术断言构造一个交互式证明系统。直观上, 我们需要提出一个交互式过程, 其中证明者与验证者进行多次交互, 每次迭代都使要证明的剩余断言变得更简单 (即至少减少一个变量)。这个交互式过程具有上述能力似乎是由于各种断言针对的是算术问题而非原始的布尔问题 (实际上, 可以说, 关键在于这些断言针对着推广的问题而非原始问题)。

教学注释: 本节用大部分篇幅证明 $\text{coNP} \subseteq IP$, 并建议在课堂上也这样做。在证明时重点介绍思想, 而忽略各种实现细节 (这些细节可见参考文献[162, 202])。

出发点

通过对不可满足的 CNF 范式集构造交互式证明系统来证明 $\text{coNP} \subseteq IP$, 此类 CNF 范式集是 coNP -完全的。因此, 我们的出发点是一个给定的不可满足的布尔 CNF 范式。

布尔 (CNF) 范式的算术化

给定一个布尔范式, 将其中的布尔变量用整数变量代替, 并将逻辑运算用相应的算术运算代替。特别地, 将布尔取值的“真”和“假”分别用整数 1 和 0 代替, OR-子句用加法代替, 而将最高一级的合取用乘法代替。图 9.1 描述了这种变换。注意: 布尔范式被某个特定

① 当 G_1 与 G_2 不同构时, 不存在与两个图都同构的图 (即既同构于 G_1 也同构于 G_2)。此时, 在步骤 (V_1) 发送的图 G' 唯一确定了 σ 。另一方面, 如果 G_1 与 G_2 同构, 则对步骤 (V_1) 发送的任何图 G' , G_1 与 G' 间的同构映射数量等于 G_2 与 G' 间的同构映射数量。从而, 在这种情况下, 从 G' 中得不到关于 σ (由验证者选择) 的任何信息, 所以证明者使验证者确信的概率不会超过 1/2。

的真值赋值满足（或不满足），当且仅当对相应的 0-1 赋值计算算术表达式时得到一个正（整数）值（或得到 0）。因此，称原始布尔范式是不可满足的，可以解释为断言所得到的算术表达式之和对于所有的 0-1 赋值取值均为 0。例如，布尔公式

$$(x_3 \vee \neg x_5 \vee x_{17}) \wedge (x_5 \vee x_9) \wedge (\neg x_3 \vee \neg x_4)$$

被替换为算术表达式

$$(x_3 + (1 - x_5) + x_{17}) \cdot (x_5 + x_9) \cdot ((1 - x_3) + (1 - x_4))$$

布尔公式不可满足当且仅当其算术表达式对所有 $x_1, x_2, \dots, x_{17} \in \{0, 1\}$ 的取值求和的结果等于 0。因此，证明原始布尔公式的不可满足性可归约为证明相应的算术和为 0。我们强调关于算术表达式还有两个结论。

(1) 算术表达式是整数上的低次多项式，具体地，其次数为原始布尔公式中的子句数量。

(2) 对任意布尔公式，相应算术表达式（关于任意 $x_1, x_2, \dots, x_n \in \{0, 1\}$ ）的取值位于区间 $[0, v^m]$ 之内，其中 v 为子句中变量的最大数目，而 m 为子句个数。因此，对所有 2^n 种可能的 0-1 赋值求和，可得到一个 $[0, 2^n v^m]$ 中的整数，其中 $n \leq vm$ 为变量个数。

	布尔表示	算术表示
变量值	假, 真	0, 1
连接符	$\neg, \vee$ 及 $\wedge$	$1-X, +$ 和
最终真值	假, 真	0, 正值

图 9.1 CNF 范式的算术化

转移至有限域中

一般情况下，要检查 $[0, M]$ 中两个整数是否相等，只需检查 $\text{mod } q$ 是否相等即可，其中 $q > M$ 。这样做的好处是，当 q 为素数时，算术运算成为有限域中的运算 ($\text{mod } q$)，从而某些事情会更“好办”（如均匀选择一个值）。因此，证明一个 CNF 范式不可满足的问题可归结为证明具有如下形式的不等式

$$\sum_{x_1=0,1} \cdots \sum_{x_n=0,1} \phi(x_1, x_2, \dots, x_n) \equiv 0 \pmod{q} \quad (9.1)$$

其中 ϕ 为低次的多变量多项式 (q 可用 $O(|\phi|)$ 个比特表示)。在命题剩余的部分，所有算术运算都在有限域 $\text{GF}(q)$ 中进行。

协议概述：通过迭代减少求和

给定一个形如式 (9.1) 的表达式，可以利用迭代来逐渐减少求和，每次迭代中去掉一次求和，并为相应的自由变量赋值。在每次迭代开始时，假设证明者提供了一个单变量多项式，将剩余表达式表示为当前被去除的（单个）变量的函数。（由结论 1 可知，这个多项式次数较低，从而具有较短的描述。）^①验证者检查该多项式（假定为 p ）是否次数较低，并对当前公布的值（假定为 v ）作出响应（即验证是否有 $p(0) + p(1) \equiv v$ ）。下一步，验证者对当前的自由变量随机赋值（即均匀选择 $r \in \text{GF}(q)$ ），得到表达式的一个新的求值（即验证者计算 $v \leftarrow p(r)$ ，并希望证明剩余表达式等于 v ）。验证者将均匀选择的赋值（即 r ）传给证明者，

^① 我们还利用了结论 2，该结论暗示可以使用一个有限域，其中元素长度是原始布尔公式长度（即 $\log_2 q = O(vm)$ ）的多项式。

然后双方进入下一次迭代（针对剩余表达式和新的 v 值）。最后一次迭代结束时，验证者有一个闭合形式的表达式（即不含求和的表达式），易检查该式是否等于期望值。

一次迭代（细节）：第 i 次迭代的目标是证明如下断言

$$\sum_{x_i=0,1} \cdots \sum_{x_n=0,1} \phi(r_1, r_2, \dots, r_{i-1}, x_i, x_{i+1}, x_{i+2}, \dots, x_n) \equiv v_{i-1} \pmod{q} \quad (9.2)$$

其中 $v_0 = 0$ ， $r_1, r_2, \dots, r_{i-1}$ 和 v_{i-1} 由前面的迭代中确定。第 i 次迭代包含两步（两个消息）：证明者步骤和验证者步骤。假设证明者向验证者提供了单变量多项式 p_i ，满足

$$p_i(z) \stackrel{\text{def}}{=} \sum_{x_{i+1}=0,1} \cdots \sum_{x_n=0,1} \phi(r_1, r_2, \dots, r_{i-1}, z, x_{i+1}, x_{i+2}, \dots, x_n) \pmod{q} \quad (9.3)$$

注意：在 $\text{mod } q$ 情况下， $p_i(0) + p_i(1)$ 等于式 (9.2) 的左边。将证明者实际发出的多项式记作 p'_i （即对诚实的证明者而言， $p'_i = p_i$ ）。验证者首先检查是否有 $p'_i(0) + p'_i(1) \equiv v_{i-1} \pmod{q}$ ，再均匀选择 $r_i \in \text{GF}(q)$ 并发送给证明者。当然，如果第一个检查未通过，则验证者会拒绝。下一次迭代中要证明的断言为

$$\sum_{x_{i+1}=0,1} \cdots \sum_{x_n=0,1} \phi(r_1, r_2, \dots, r_{i-1}, r_i, x_{i+1}, x_{i+2}, \dots, x_n) \equiv v_i \pmod{q} \quad (9.4)$$

其中 $v_i \stackrel{\text{def}}{=} p'_i(r_i) \pmod{q}$ 由各方分别计算。

协议的完备性：当初始断言（即式 (9.1)）成立时，证明者会提供正确的多项式（由式 (9.3) 确定），从而使验证者总是接受断言。

协议的合理性：对于某次特定的迭代，规定起始断言（即式 (9.2)）错误而终止断言（即式 (9.4)）正确的概率上限就足够了。讨论第 i 次迭代，令 v_{i-1} 和 p_i 分别满足式 (9.2) 和式 (9.3)，即 v_{i-1} 为第 i 次迭代开始时公布的（错误）值，而 p_i 表示去掉当前变量（如式 (9.3)）后得到的多项式。令 $p'_i(\cdot)$ 为证明者的可能应答。不失一般性，可以假设 $p'_i(0) + p'_i(1) \equiv v_{i-1} \pmod{q}$ 且 p'_i 次数较低（否则，验证者一定会拒绝）。利用假设（即起始断言式 (9.2) 为假），可知 $p_i(0) + p_i(1) \not\equiv v_{i-1} \pmod{q}$ 。从而， p'_i 和 p_i 是不同的低次多项式，它们取值相同的点极其稀少（如果有）。现在，如果验证者的赋值（即其随机选择的 r_i ）不是这些使两个多项式取值相同的点（即 $p_i(r_i) \not\equiv p'_i(r_i) \pmod{q}$ ），则本次迭代的终止断言（即式 (9.4)）也为假（因为新的值（即 v_i ）已经被设置为等于 $p'_i(r_i) \pmod{q}$ ），而剩余表达式等于 $p_i(r_i)$ 。细节留作练习（见习题 9.3）。

这就证明了不可满足的 CNF 范式集合具备交互式证明系统。实际上，可以利用一个类似的证明系统（利用相关的算术——见习题 9.4）来证明给定公式具有给定数量的可满足性赋值，即证明（“可数”）集合

$$\{(\phi, k) : |\{\tau : \phi(\tau) = 1\}| = k\} \quad (9.5)$$

的成员问题。

利用充分的归约，可以证明 $\#P$ 中任何问题都具备交互式证明系统（即对任意 $R \in PC$ ，集合 $\{(x, k) : |\{y : (x, y) \in R\}| = k\}$ 属于 IP ）。证明 $PSPACE \subseteq IP$ 需要更多的工作，以下将分别介绍。

PSPACE 类的交互式证明 (基本思想)

我们提出对可满足的量化布尔公式 (Quantified Boolean Formulae, QBF) 的交互式证明, QBF 在 $PSPACE$ 中是完全的 (见定理 5.15)。①注意: 在这种公式中对量词数量没有限制 (可以是输入长度的多项式), 并且既有存在量词又有全称量词, 而这些量词还可以变化。在这种公式的算术化表达中, 将存在量词用求和代替, 全称量词用乘积代替。然而在对得到的算术表达式使用上述协议时, 会遇到两个困难。首先表达式的 (内部) 值 (可能包含大量嵌套式乘积) 的上限只由一个 (关于输入长度的) 双指数函数确定。其次, 当拆分一个和 (或乘积) 时, 表达式可能为一个高次多项式 (由于嵌套乘积可能出现在剩余表达中)。②例如, 在如下表达式中, 两种现象均出现。

$$\sum_{x=0,1} \prod_{y_1=0,1} \cdots \prod_{y_n=0,1} (x + y_n)$$

上式等于 $\sum_{x=0,1} x^{2^{n-1}} \cdot (1+x)^{2^{n-1}}$ 。第一个困难可利用以下事实 (将在习题 9.7 中证明) 解决, 即如果 $[0, M]$ 中的两个整数是不同的, 则对于区间 $[3, \text{poly}(\log M)]$ 中大部分素数, 它们取模后也都不相同。因此, 让验证者随机选择素数 q , 长度为原始公式长度的线性函数, 则证明双方只需考虑模 q 之后的算术表达式。解决第二个困难时, 考虑 $PSPACE$ 实际上可归约为 QBF 的一种 (非正则) 特殊形式, 其中任何变量都不会同时出现在多于一个全称量词的左边和右边 (见定理 5.15 的证明或习题 9.6)。从而, 当对结果算术表达式中的和 (或乘积) 进行算术化以及拆分时, 相应的单变量多项式次数较低 (即最多为原始公式长度的 2 倍, 之所以是 2 倍, 是因为单个全称量词对于该变量在左边赋值或在右边赋值)。

IP 包含于 PSPACE 中

我们将证明任意交互式证明系统存在一个最优的证明者策略, 可在多项式空间内实现, 这里最优的证明者策略是指使指定验证者接受公共输入的概率最大的一种策略。从而 $IP \subseteq PSPACE$, 因为 (对任意 $S \in IP$) 可在多项式空间内模仿指定验证者与具有固定多项式空间证明者策略的所有可能交互。

命题 9.5 设 V 为概率多项式时间 (验证者) 策略, 则存在一种多项式空间计算的 (证明者) 策略 f , 使得对任意 x , V 接受 x 的概率最大。即对任意 P^* 和 x , V 与 P^* 交互后接受 x 的概率上限为 V 与 f 交互后接受 x 的概率。

证明概要: 对任意公共输入 x 及到某一时刻任意可能的部分交互副本 γ , 策略^③ f 通过穷举验证者使用的与 (x, γ) 一致的所有可能的内部随机数, 可以确定对证明者来说最优的下一条消息。具体地, f 通过递归方法来确定, 使得若 m 为验证者使用的与 (x, γ) 一致的随机数的最大数量时, 则令 $f(x, \gamma) = m$, 且当后续的证明者行为由 f 确定时 (这是使用递归的原因), 验证者接受输入 x 。即如果以下两个条件成立, 则验证者的随机序列 r 支持环境 $f(x, \gamma) = m$, 其中 $\gamma = (\alpha_1, \beta_1, \dots, \alpha_t, \beta_t)$ 。

① 实际上, 以下对于上述证明系统的延展可得到不可满足量化布尔公式集合 (在 $PSPACE$ 中仍为完备的) 的证明系统。另外, 通过对习题 9.5 中的证明系统进行扩展, 可得到 QBF 的另一种交互式证明系统。

② 这个较高的次数会造成两个困难, 但只有第二个困难比较严重。第一个困难是相应协议的合理性要求工作于有限域上, 且该域远远大于这个高的次数, 这是可以做到的 (因为次数最多是公式长度的指数)。第二个困难 (更严重的一个) 是多项式可能有大量的 (指数个数的) 非 0 系数, 从而验证者无法读出该多项式的标准表达式 (以所有非 0 系数列表的形式)。事实上, 某些情况下该多项式可能存在其他简洁和高效的表式 (如以下例子中), 但是尚不清楚一般情况下如何得到这种表示。

③ 为便于讨论, 在描述策略 f 时, 使用与 V 交互的整个部分副本 (而不仅仅是 V 之前发出的消息序列)。

(1) r 与 (x, γ) 一致, 这意味着对任意 $i \in \{1, 2, \dots, t\}$, 有 $\beta_i = V(x, r, \alpha_1, \alpha_2, \dots, \alpha_i)$ 。

(2) 当证明者的后继行为由 f 确定时, r 使 V 接受, 这意味着在结束时 (T 轮交互之后), 有

$$V(x, r, \alpha_1, \alpha_2, \dots, \alpha_t, m, \alpha_{t+2}, \alpha_{t+3}, \dots, \alpha_T) = 1$$

其中对任意 $i \in \{t+1, t+2, \dots, T-1\}$, 有 $\alpha_{i+1} = f(x, \gamma, m, \beta_{t+1}, \beta_{t+2}, \dots, \alpha_i, \beta_i)$, 且 $\beta_i = V(x, r, \alpha_1, \alpha_2, \dots, \alpha_i, m, \alpha_{i+1}, \alpha_{i+2}, \dots, \alpha_i)$ 。

因此, 如果 m 使 $E[\xi_{f,V}(x, R_\gamma, \gamma, m)]$ 的值最大, 则 $f(x, \gamma) = m$, 其中 R_γ 在与 (x, γ) 一致的 r 中均匀选择, 且 $\xi_{f,V}(x, r, \gamma, m)$ 指示着 V 在与 f 的后续交互中是否接受 x (关于随机数 r 和部分副本 (γ, m))。从而, 当给定对 $f(x, \gamma, \cdot)$ 的预言访问时, $f(x, \gamma)$ 的值可在多项式空间内计算。通过对空间受限计算的标准组合 (即将独立空间分配给递归中的每一层, 并在所有递归中对同一层的调用使用相同的空间), 命题得证。□

9.1.4 变形及更好的结构: 概述

本小节讨论交互式证明基本概念的几种变形, 以及更精细的复杂性量度。本节的内容较前沿, 只简要介绍了各种概念和结论 (并指出结论证明的出处)。

9.1.4.1 Arthur-Merlin 游戏 (公开掷硬币证明系统)

在通用的交互式证明系统中, 验计者基于其在交互中的观察来自行 (但高效地) 确定要发出的消息 (观察中包括验证者使用的随机数, 不失一般性, 假设其在交互开始时已生成)。因此, 验证者以往使用的随机数不一定包含在其发送的消息中。相反, 在公开掷硬币证明系统中 (又称 Arthur-Merlin 证明系统), 验证者的消息里包含了其在本轮交互中生成的随机数。因此, 消息中揭示了用于生成这些消息的随机数 (这些随机数被公开了)。不失一般性, 验证者的消息可以与本轮交互中生成的随机数相同 (因为验证者基于这些随机数而求出的其他串实际上也是由它们确定的)。

注意: 定理 9.4 证明中提出的证明系统即属公开掷硬币型, 而非同构图的证明系统 (构造 9.3) 则不在此列。因此, 虽然并非所有的证明系统都属于公开掷硬币型, 但由定理 9.4, 任意具有交互式证明系统的集合也具有公开掷硬币的交互证明系统。这就意味着, 在交互式证明系统环境下, 随机提出问题与 “巧妙地” 提出问题具有相同的功能。(在 9.1.4.3 节结尾我们会给出更强的结论。)

事实上, 公开掷硬币系统是一种有限制条件的交互式证明系统。该条件使系统的设计更加复杂, 但却便于分析 (特别是针对一般系统的分析)。公开掷硬币证明系统的另一个优点是, 验证者在确定行为时 (除最后一次判定外) 可以无视证明者消息。定理 9.12 的证明将利用这一性质。

9.1.4.2 具有双边错误的交互式证明系统

定义 9.1 的合理性条件中允许错误概率, 但完备性条件中不允许。此时, 称证明系统具有完善的完备性 (或单边错误概率)。更通用的定义则在完备性和合理性条件中都允许错误概率 (如上限为 $1/3$)。注意: 具有这种推广的 (双边错误) 交互证明集合也包含于 $PSPACE$ 中, 因此允许双边错误并不会使交互式证明更加强大。见 9.1.4.3 节最后的进一步讨论。

9.1.4.3 交互式证明系统的分层

定义 9.1 只涉及验证者的全部计算时间, 从而允许任意 (多项式) 数量的消息交换, 而更精细的定义还需规定交换的消息数量 (也称为轮数) ^①。

定义 9.6 (交互式证明的轮复杂度)

• 对于整数函数 m , 复杂性类 $IP(m)$ 包含的集合具有交互式证明系统, 且在证明中对于公共输入 x , 双方最多交换 $m(|x|)$ 个消息。 ^②

• 对于整数函数集 M , 令 $IP(M) \stackrel{\text{def}}{=} \bigcup_{m \in M} IP(m)$, 则 $IP = IP(\text{poly})$ 。

例如, 如果验证者只发出一个消息, 而证明者也只回答一个消息, 则这类交互式证明系统对应于 $IP(2)$ 。显然, $NP \subseteq IP(1)$, 且可能是严格的包含关系, 因为在 $IP(1)$ 中, 验证者收到证明者发出的单个消息后可能还会生成随机数 (还要注意的, $IP(0) = \text{co}RP$)。

定义 9.6 还引出对交互式证明系统的分层, 其中不同的“层”对应于系统轮复杂度的不同“增长率”。关于分层有如下的已知结论。

• 线性加速 (见附录 F.2 (或参考文献[23]及参考文献[111]))。设整数函数 f 满足对所有 n 有 $f(n) \geq 2$, 则 $IP(O(f(\cdot)))$ 类可退化为 $IP(f(\cdot))$ 类。特别地, $IP(O(1))$ 退化为 $IP(2)$ 。

• $IP(2)$ 类中包含了已知不属于 NP 的集合, 如非同构图 (见构造 9.3)。然而, 在合理的困难性假设下, $IP(2) = NP$ (见参考文献[167])。

• 如果 $\text{co}NP \subseteq IP(2)$, 则多项式时间层级也会退化 (见参考文献[45])。

通常, 人们猜想 $\text{co}NP$ 不包含于 $IP(2)$ 中, 所以具有无穷多消息交换的交互式证明比消息数量受限 (如规定为常数) 的证明更强大。 ^③

$IP(1)$ 类, 也记作 $\mathcal{MA}$, 似乎是 NP 类“真正”的随机化版本 (然而是非交互式的): 其中证明者提供一个候选的 (多项式大小) “证明”, 验证者概率地评估其正确性 (而非确定地)。

IP 层级 (即 $IP(\cdot)$) 等价于一种类似的层级, 记作 $\mathcal{AM}(\cdot)$, 它针对着公开掷硬币 (即 Arthur-Merlin) 交互式证明。对任意整函数 f , 有 $\mathcal{AM}(f) = IP(f)$ 。当 $f \geq 2$ 时, 还有 $\mathcal{AM}(f) = \mathcal{AM}(O(f))$ 。实际上, 上述对 $IP(\cdot)$ 的线性加速可结合以下两个结论来证明。

(1) 用 $\mathcal{AM}(\cdot)$ 模仿 $IP(\cdot)$ (见附录 F.2.1 或参考文献[111]): $IP(f) \subseteq \mathcal{AM}(f+3)$ 。

(2) 对 $\mathcal{AM}(\cdot)$ 线性加速 (见附录 F.2.2 或参考文献[23]): $\mathcal{AM}(2f) \subseteq \mathcal{AM}(f+1)$ 。

特别地, $IP(O(1)) = \mathcal{AM}(2)$, 即使是对 $\mathcal{AM}(2)$ 做出限制, 让验证者接收到证明者消息后不再生成随机数, 等式也成立 (注意: $IP(1) = \mathcal{AM}(1)$ 和 $IP(0) = \mathcal{AM}(0)$ 是两种平凡情况)。通常, 可将 $\mathcal{AM}(2)$ 简写为 $\mathcal{AM}$, 但这和将 IP 作为 $IP(\text{poly})$ 的缩写并不一致。

$IP(O(f)) = IP(f)$ 可用 $\mathcal{AM}(\cdot)$ 中的类似结论来证明, 它表明了公开掷硬币环境下研究交互式证明的优势。在证明 IP -分层等价于类似的双边误差分层时, 也有类似现象 (见习题 9.8)。

9.1.4.4 独到之处

我们强调, 虽然已经放宽了对验证程序的要求 (允许其与证明者交互, 生成随机数, 以

① 更复杂的结构中还需考虑证明者发出消息的总长度 (见参考文献[106])。

② 只对交换的消息计数而不考虑通信方向。

③ 注意: 线性加速不能无限次地使用, 因为每次应用都使验证的时间复杂度增加 (如变为原来的平方)。

及某种（有限的）错误概率），但没有用以限制验证者判决合理性的证明者假设。这与其他证明系统截然相反，如在计算合理的证明（见 9.1.5.2 节）中，验证者判决的合理性依赖于对证明者的假设条件。

9.1.5 计算能力受限的证明者：概述

回顾交互式证明的定义（即定义 9.1）中并未规定证明者的计算能力。这一事实会导致两个矛盾的结果。

(1) 完备性条件中没有规定相应证明策略（使验证者接受正确断言）的复杂度上限。

(2) 合理性条件则保证，在不考虑欺骗性证明者的计算时，验证者不能被骗而接受不正确的断言（接受概率超过合理性误差中的概率）。

注意：在交互式证明系统 (P, V) 中，规定证明者策略 P 的复杂度上限必定会强化 (P, V) 是相应（正确断言）集合的证明系统这一断言。我们强调预先制定的证明者策略只针对完备性条件（与合理性条件无关）。另一方面，放宽交互式证明的定义，使合理性只在某一类特定的欺骗性证明者策略中成立（而非所有的欺骗性证明者策略），将弱化相应的断言。本小节考虑上述两种情况。

教学注释：本小节介绍前沿性知识，最好留给学生独立阅读。其中只有各种概念的简介，读者可参考章节注释，以了解更多细节（通过参阅相应文献）。

9.1.5.1 证明者有多强大

假设集合 S 属于 IP ，这意味着存在验证者 V ，一定会接受来自 S 的输入，且不能被欺骗（除以极小概率外）而接受任何不属于 S 的输入。人们可能会问，究竟证明者需要具备何种能力才能使验证者 V 一定接受 S 中的输入。注意：命题 9.5 中称最优的证明者策略（使任意验证者 V 确信）可在多项式空间内实现，并且不能期望对于 $PSPACE=IP$ 中的集合有更好的空间复杂度（这是因为模仿 V 与最优证明者策略的交互将得到对集合的判定程序）。然而，对于使特定验证者确信的某个证明者策略，我们仍可寻求其复杂度的更好上限，其中特定验证者对应于特定的集合 S 。更有趣的是，考虑所有发起针对 S 的交互式证明的验证者，我们希望达到足以使这些验证者中任意一个确信（接受 S 中的任意输入）所需的计算能力上限。

在这里要强调，不同于计算上合理的证明系统（见 9.1.5.2 节），在合理性条件中不限制证明者能力，但是会考虑符合完备性条件的证明者的最小复杂度。具体而言，我们感兴趣的是符合完备性条件的相对高效的证明者。“相对高效的证明者”有 3 种不同解释，如下所述。

(1) 如果一个证明者在给定辅助输入（除了来自于 S 中的公共输入外）时，运行于（概率）多项式时间，则认为它是相对高效的。具体地，当 $S \in NP$ 时，辅助输入可能是该公共输入属于 S 的 NP-证据。然而，即便在这种情况下，也不需要证明者将辅助输入发送给验证者，如可以设计证明者为零知识的（见构造 9.10）。

这种条件是充分的，有些情况下无辅助输入时交互双方将不再是多项式时间，此时，该条件十分关键。典型地，在密码学中这种辅助输入较常见，其中双方希望（零知识地）证明他们正确执行了某种计算。此时，NP-证明只是其用以生成结果的计算的一个副本，而证明者一方可以得到辅助输入。

(2) 如果证明者可以用能访问集合 S 本身的概率多项式时间预言机实现，则认为它是相对高效的。注意：构造 9.3 中的证明者具有此种性质（还可见于习题 9.10）。

这种解释推广了 NP-证明系统的自归约性。回顾 NP-集（或相应的 NP-证明系统）的自归约性是指寻找 NP-证据的搜索问题可在多项式时间内归约为集合成员的判定问题（见定义 2.14）。这里要求（相应的交互证明中）实现证明者策略可在多项式时间内归约为集合成员判定。

(3) 如果证明者可以用集合成员判定的多项式时间概率机实现，则认为它是相对高效的。这种解释把使一个“懒验证者”确信所需的时间复杂度与寻找真相（即判定集合成员）的复杂度相联系。

因此，不同于第一种解释适用于断言与其 NP-证据相伴产生的环境，第三种解释适用于只向证明者提供断言，而证明者（在与一个懒验证者交互使其确信之前）必须自己寻找证明的环境。

9.1.5.2 计算上的合理性

对合理性条件适当放宽，只考虑用高效的方法尝试欺骗验证者（并非尝试所有方法），就得到一种完全不同的证明系统。在这种系统中，验证者的判决不一定正确，但当欺骗性证明者不超过预设的复杂度上限时却相当合理。如 9.1.5.1 节所述，“相对高效”的概念可以有不同解释，最常见的一种是欺骗性证明策略可以用（非一致的）多项式规模电路实现。这符合 9.1.5.1 节中的第一种解释（即给定了（多项式长度）辅助输入的概率多项式时间策略）。在这种情况下，合理性条件可用如下的计算合理性条件取代，其中称不可能欺骗验证者接受错误断言。形式化地描述为：

对任意可用多项式规模电路簇 $\{C_n\}$ 实现的证明者策略，以及任意足够长的 $x \in \{0,1\}^* \setminus S$ ， V 与 $C_{|x|}$ 交互时接受 x 的概率小于 $1/2$ 。

在标准的合理性条件中，计算上合理的误差可以通过重复执行来减小。但是要注意，虽然如此，与标准合理性条件（其中支持串行和并行的重复）不同，并行重复并不总是减少计算合理的误差。

在完备性和合理性条件中，证明者策略都满足同样的相对高效性，这很正常。根据上述解释，满足这些条件的协议被称为论据（Arguments）。我们指出，论据系统比交互式证明系统更高效（如用通信复杂度衡量时）。

9.2 零知识证明系统

人们认为标准的数学证明给出了（额外的）知识，而不仅仅是证明断言的正确性。也就是说，通常认为（好的）证明有助于更深入地理解要证明的定理。在技术层面上，从对某个集合 $S \in \mathcal{NP} \setminus \mathcal{P}$ 的成员判定的 NP-证明中，可以得到难以计算（即使假设输入属于 S 时）的东西（即 NP-证明本身）。例如，图的 3-着色方案构成了 3-着色问题的 NP-证明，但是还给出了一些似乎难以计算（给定任意一个 3-着色图时）的信息（即着色方案）。

零知识证明中一个自然的问题是证明一个断言是否总需要给出某种额外的知识。在交互式证明系统环境中，对这个问题的回答可以是否定的：与能得到大量知识的 NP-证明相反，（交互式的）零知识证明根本得不到任何知识。即零知识证明可以使人确信，但是除了断言的正确性之外，得不到任何知识。例如，从图的 3-着色性的零知识证明中不会得到关于该图难以计算的任何信息（如 3-着色方案的部分信息）。因此，零知识证明在使人确信（断言正确性）与传递除断言正确性之外的任何信息之间形成了极端的对比。

当然，零知识证明的概念非常吸引人（如因为区分了证明—验证与学习）。然而，读者会质疑它是否有价值，因为在许多环境下，我们确实希望尽量多得到些知识。然而，在其他环境中（最著名的是在密码学中），确实需要限制另一方从证明中得到的信息（特别是，在使其确信断言正确性之外，应令其得到的信息最少）。零知识证明在密码学中的应用十分广泛，典型地，它们被当作一个工具来迫使（可能为恶意的）通信各方遵守某种预先确定的协议（而不泄露其自身的秘密输入）。有兴趣的读者可以参考附录 C.4.3.2 和附录 C.7.3.2 中的讨论（并见参考文献[91, 92]中的细节）。我们还要指出，除了直接应用于密码学外，零知识证明还为密码协议中各种问题的研究提供了参照（图 9.2）。

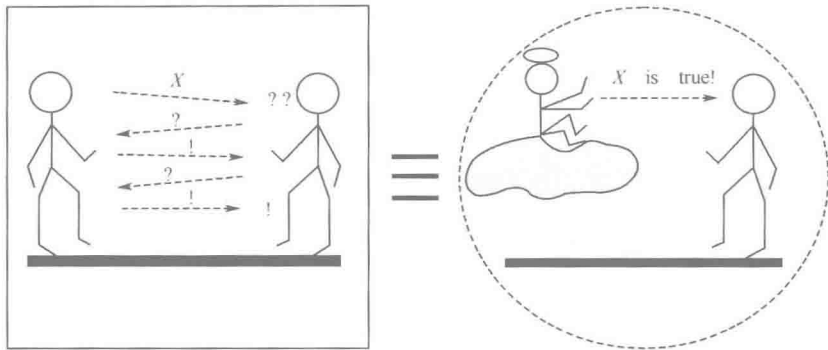


图 9.2 零知识证明——图示

教学注释：我们认为本节讲述的零知识证明在复杂性理论的课程中是足够了。关于零知识证明更详细的内容，有兴趣的读者可以见参考文献[91]中第4章。

9.2.1 定义

零知识证明在证明断言的正确性之外不提供任何信息，即验证者在证明中只能确信断言的正确性。这可以形式化地说成从零知识证明中高效获取的任何东西都可由（正确）断言本身获得。定义采用了仿真模式，以下详细讨论。

9.2.1.1 更广阔的视角：仿真模式

在定义零知识证明的定义中，将验证者看作一个潜在的敌手，尝试从（指定）证明者那里获取某种知识。^①我们要声明，对于验证者而言，不存在（可行的）敌手策略能使其从证明中获取知识（除了确信断言的正确性以外）。问题在于，如何刻划“无法获取”的条件。

让我们从更普遍的角度考虑。这里安全模型中的关键问题是如何表示直觉上敌手通过偏离诚实的预定行为而仍“无法获取任何信息”。可以这样解答：如果敌手通过无限制行为获得的信息也能由温和（或预先规定的）行为以同样的计算代价获得，则称敌手没有得到任何信息。这里“温和行为”解释了我们希望达到的安全性目标，它与研究的具体安全问题相关。例如，在零知识证明中，温和行为是任何（仅）基于断言本身的计算（并假设该断言是正确的）。因此，零知识证明是这样一种交互式证明，其中不存在可行的恶意验证者策略，使验证者从交互中得到的信息比（相信断言的）“温和参与方”得到的信息更多。

^① 回顾在定义证明系统（如交互式证明系统）时，我们将证明者视为一个潜在的敌手，要欺骗（预定）验证者（接受不正确断言）。

上述将“得不到任何信息”解释为：任意可行的敌手行为可以用温和行为来“模仿”（从而前者得不到任何信息）。这种推理方法称为仿真模式，它在密码学许多定义中起着关键作用（如它是定义加密方案和密码协议安全性的基础），进一步的细节可参考附录 C。

9.2.1.2 基本概念

回到零知识的定义。首先要指出，零知识是证明者策略具有的性质，更一般地，零知识是某些策略的属性。给定任意一个策略（如一个指定的证明者），我们考虑任意一个可行的敌手（如验证者）与上述给定策略关于来自指定集合（此时指正确断言集合）的公共输入交互后，所获取（或计算得到）的信息。此时，可以比较得到的信息与由任意可行算法（称为模拟器）对给定输入求出的信息。如果（本质不同的环境中）两种“计算能力”是等价的，则称上述给定策略为零知识的。具体细节如下。

零知识条件的定义针对两种类型的概率空间，每个空间将概率分布与相关输入（如一个正确断言）关联。具体地，在交互式证明中，第一种概率空间表示验证者在与特定证明者策略 P （关于某个公共输入）交互后的输出分布，这里验证者可以使用任意一种高效策略（不一定是规定的那个）。第二种概率空间表示某种概率多项式时间算法（在只给定相应输入（且不与任何人交互）时）的输出分布。零知识的基本模型称对任意一个第一种空间，存在一个“类似的”第二种空间。根据对相似性的不同理解，在具体变量上有所区别。最严格的一种理解就是完全零知识的概念，此时的相似性即是相等。

定义 9.7（完全零知识，一种过于简单的定义）^① 集合 S 上的证明者策略 P 称为完全零知识的，如果对任意概率多项式时间验证者策略 V^* ，存在概率多项式时间算法 A^* ，满足对任意 $x \in S$ ，有

$$(P, V^*)(x) \equiv A^*(x)$$

其中 $(P, V^*)(x)$ 是一个随机变量，表示与证明者 P 对于公共输入 x 进行交互后，验证者 V^* 的输出，而 $A^*(x)$ 表示算法 A^* 对于输入 x 计算结果的随机变量。

我们指出， coRP 中任意集合均有完全零知识证明系统，其中证明者保持沉默，验证者自己进行判断即可。同样在 BPP 类中也成立，条件是放宽对交互式证明系统的定义，使其允许双边错误。当然，这里的重点是非平凡证明系统，即 BPP 类以外集合的证明系统。

进一步放宽对相似性概念的理解，可以得到几乎完全的零知识（也称统计零知识），其中的相似性意味着统计接近（即概率空间之差可忽略）。而（对相似性）最自由的解释，就是通常使用的标准术语零知识（有时指计算上的零知识），其中相似性指计算不可区分性（即不存在高效程序来区分两种概率空间）。将上述讨论与计算不可区分性的定义（即附录 C.3 中的定义 C.5）相结合，得到如下定义。

定义 9.8（零知识，有些简单的定义） 集合 S 上的证明者策略 P 被称为是零知识的，如果对任意概率多项式时间的验证者策略 V^* ，存在一个概率多项式时间模拟器 A^* ，使得对任意概率多项式时间的区分器 D ，函数

$$d(n) \stackrel{\text{def}}{=} \max_{x \in S \cap \{0,1\}^n} \left\{ \Pr \left[D(x, (P, V^*)(x)) = 1 \right] - \Pr \left[D(x, A^*(x)) = 1 \right] \right\}$$

^① 在实际定义中，可以用如下两种方式放宽要求：第一种方法允许 A^* 运行于期望（而非严格规定）的多项式时间；第二种方法允许 A^* 以不超过 1/2 的概率不输出任何值，并只有在有输出时才考虑其输出值。第二种方法蕴含了第一种，但反之是否成立尚且未知。

是可忽略的。^①用 ZK 表示具有交互式零知识证明系统的集合类。

定义 9.8 是附录 C.4.2 中定义的一个简化版本。具体地,为了确保串行组合下的零知识性,必须对定义进行适当修改(为 V^* 和 A^* 提供相同的 $\text{poly}(|x|)$ -比特辅助输入)。附录 C.4.4 中讨论了其他一些定义问题及相关概念。

随机性和交互的作用

可以证明,只有 BPP 中的集合才具备确定型验证者的零知识证明(见习题 9.13)。对于确定型证明者有同样结论,条件是要考虑“辅助输入”的零知识性(如定义 C.9)。还可以证明,只有 BPP 中的集合才具备只发送单个消息的零知识证明(见习题 9.14)。因此,随机性和交互在非平凡的零知识证明系统中起着重要作用(进一步的细节可见参考文献[91]中 4.5.1 节)。

高级注释:知识复杂性

知识复杂性层级对“交互中揭示的知识”进行量化,而零知识是知识复杂性层级中的最低层次。具体来讲,交互式证明系统的知识复杂性定义为高效模仿与证明者的交互时所需预言询问的最小数量(见参考文献[90]中 2.3.1 节)。

9.2.2 零知识证明的功能

面对一个像零知识这样复杂(且似乎自相矛盾)的定义时,人们可能会疑惑定义中的条件是否能真正满足(以非平凡方式)。^②人们发现非平凡零知识证明的存在性与 NP 中难解问题的存在性相关。特别地,本小节将证明如果存在单向函数,则每个 NP -集都有零知识证明系统(逆命题的证明可见参考文献[91]中 4.5.2 节或参考文献[228])。但是,首先要对某个特定的不属于 BPP 类的 NP 集,给出一个简单的(完全)零知识证明系统,以此来阐述零知识证明的非平凡性。在这里没有给出任何难解性假设(然而,只有当 NP 不包含于 BPP 类时该结论才有意义)。

9.2.2.1 一个简单例子

史诗《奥德塞》中漏掉了埃埃亚(Aeaea)岛上的迷宫问题。女巫喀耳刻,太阳神赫利俄斯的女儿,向天神一般的奥德修斯提出挑战,要求他从北门到南门横穿迷宫。奥德修斯质疑这样的通道是否存在,并要求喀耳刻证明。喀耳刻回答,如果指出一条通道,就无法进行挑战了。“不要紧,”聪明的奥德修斯说,“你可以使用魔法将我放到迷宫中任意一点,我任选一个门,你指给我从那个门出去的路。如果能多次重复这个实验,我就相信两个门之间确实存在一条通道,而你也没有向我提示这样的通道”。“实际上,”聪明的喀耳刻自言自语,“为这个凡人指出从任意点到他选择的门的一条路径,确实不会让他知道除了这条路之外更多的信息。”

上面的故事描述了图同构问题的零知识证明的主要思想。前面提到所有同构的图对构成的集合不属于 BPP 类,因此直接的 NP -证明(证明者提供同构映射)并非零知识的。进一

① d 的衰减速度大于任意正多项式的倒数(即对任意正多项式 p 及足够大的 n , 有 $d(n) < 1/p(n)$)。当然,如果 $S \cap \{0,1\}^n = \emptyset$, 则 $d(n) \stackrel{\text{def}}{=} 0$ 。

② 回顾 BPP 类中任意集合均有一种平凡的(双边错误)零知识证明系统,其中验证者自身确定成员问题。因此,关键在于 BPP 类之外的集合是否存在零知识证明。

步, 假设图的同构问题不属于 BPP 类, 则该集合不存在零知识的 NP-证明系统。然而, 我们很快将看到, 这个集合确实具有交互式零知识证明系统。

构造 9.9 (图同构问题的零知识证明)

- 公共输入。一对图 $G_1=(V_1, E_1)$ 和 $G_2=(V_2, E_2)$ 。如果两个图同构, 令 ϕ 为两者之间任意一个同构映射, 则 ϕ 在顶点集 V_1 与 V_2 间构成一一映射, 且满足 $\{u, v\} \in E_1$ 当且仅当 $\{\phi(u), \phi(v)\} \in E_2$ 。

- 证明者的第一步 (P_1)。证明者随机选择 G_2 的一个同构副本并发送给验证者。也就是说, 证明者随机均匀地选择顶点集 V_2 上的置换 π , 构造出一个图, 顶点集为 V_2 , 边集为

$$E \stackrel{\text{def}}{=} \{\{\pi(u), \pi(v)\} : \{u, v\} \in E_2\}$$

证明者将 (V_2, E) 发送给验证者。

- 动机说明。如果输入的两个图如证明者所称, 是同构的, 则 P_1 中发送的图与两个输入图均同构。然而, 如果输入图不同构, 则不存在与两者均同构的图。

- 验证者的第一步 (V_1)。收到证明者传来的图 $G'=(V', E')$ 之后, 验证者要求证明者出示 G' 与输入图之一的同构映射。即验证者随机选择 $\sigma \in \{1, 2\}$, 发送给证明者 (并希望其返回 G_σ 与 G' 间的同构)。

- 证明者的第二步 (P_2)。如果验证者传来的消息 $\sigma=2$, 则证明者将 π 发送给验证者; 否则 (即 $\sigma \neq 2$ 时), 证明者向验证者发送 $\pi \circ \phi$ (即 π 与 ϕ 的合成, 定义为 $\pi \circ \phi(v) \stackrel{\text{def}}{=} \pi(\phi(v))$)。

(事实上, 证明者将任意 $\sigma \neq 2$ 的情形都视为 $\sigma=1$ 。因此, 在分析中, 我们将不失一般性地假设 $\sigma \in \{1, 2\}$ 总成立。)

- 验证者的第二步 (V_2)。如果证明者传来的消息, 记作 ψ , 是 G_σ 与 G' 间的同构, 则验证者输出 1, 否则, 输出 0。

构造 9.9 中的验证者策略易在概率多项式时间内实现。如果给了证明者输入图间的一个同构作为辅助输入, 则证明者策略也可在概率多项式时间内实现。上述的动机说明证明构造 9.9 确实构成同构图的交互式证明系统。因此, 我们重点证明该构造的零知识性。

首先考虑一种特殊情况, 其中验证者执行一种预定策略 (随机选择 σ 而无视于收到的图 G')。验证者的观察易通过随机选择的 σ 和 ψ 来模仿, (利用映射 ψ) 构造 G_σ 的一个随机同构副本 G' , 并输出三元组 (G', σ, ψ) 。事实上 (即使在这种情况下), 模拟器的行为不同于预定证明者 (证明者是利用映射 π , 随机选择 G_2 的一个同构副本 G'), 但其输出分布与验证者在实际交互中的观察分布相同。然而, 上述讨论假设验证者执行了预定策略, 而一般情况下, 验证者可能 (敌意地) 根据收到的 G' 来选择 σ 。因此, 需要一个更加复杂的模拟 (如下所述)。

这里需要澄清, 我们希望模拟任意一个验证者策略与预定证明者间的交互。因此, 模拟器必须依赖于相应的验证者策略, 而事实上, 应该根据一个一般的验证者策略来描述模拟器。严格地说, 这意味着模拟器程序是相应验证者策略的具体化。以下的模拟器将通用的验证者策略作为子程序。

回到构造 9.9 这个具体协议, 其基本思路是模拟器试图猜测 σ , 且在猜对时结束模拟。具体地, 模拟器均匀选择 $\tau \in \{1, 2\}$ (并希望验证者后来选择的 $\sigma=\tau$), 然后通过对 G_τ 进行随机置换构造图 G' (因此, 能提出 G_τ 与 G' 间的一种同构)。注意: 只分析 “是” 实例 (即输入

的 G_1 和 G_2 为同构图)下的模拟器。要点是如果 G_1 和 G_2 同构, 则从图 G' 中得不到任何关于模拟器猜测 (即 τ) 的信息。^①因此, 恶意验证者选择的 σ 可能依赖于 G' , 但不会依赖于 τ , 这蕴含了 $\Pr[\sigma = \tau] = 1/2$ 。换言之, 模拟器猜对 (即其猜测的 $\tau = \sigma$) 的概率为 $1/2$ 。现在如果模拟器猜对了, 则其输出有正确的分布, 否则, 就重复整个过程。

总结: 几个有用的习惯

我们强调 3 个习惯, 或者 (不明显地) 用于上述分析中, 或者可用于简化其他的零知识模拟器。

(1) 不失一般性, 假设欺骗性的验证者策略可以用确定的多项式规模电路 (或等价地, 用一个带辅助输入的确定的多项式时间算法) 实现。^②

通过固定验证者生成的随机数, 并观察到对各种剩余确定性策略的 (“一致性”) 模拟中能得到对原始概率策略的模拟, 由此证明这个假设是合理的。事实上, 合理性依赖于模拟器针对的是具有任意 (多项式长度) 辅助输入的验证者。

(2) 不失一般性, 只需考虑这样的欺骗性验证者: 验证者 (只) 输出其对于交互的观察 (即公共输入, 内部产生的随机数, 以及接收到的消息)。换言之, 只需模拟欺骗性验证者在真正交互中的观察就足够了。

这种假设是合理的, 注意到, 任意验证者的最终输出可由其观察得到, 而这种转换的复杂度上限就是验证者策略的复杂度。

(3) 不失一般性, 只需构造一个 “弱模拟器”, 能以某种不可忽略的概率^③生成输出, 且无论生成何种输出, 都具有 “正确的” 分布 (即与预定证明者在真正交互中得到输出的分布相同)。

通过多次重复 (多项式次) 调用这种弱模拟器, 并利用任意一次调用中产生的第一个输出, 可证明这样做的合理性。注意: 在足够次数的调用之后, 我们无法以可忽略的概率生成一个输出。进一步地, 一个无法以可忽略概率生成输出的模拟器可以转化为总是生成输出的模拟器, 且输出分布的统计偏移可忽略。

9.2.2.2 零知识证明的全部功能

构造 9.9 中的零知识证明系统针对一个特殊的、不属于 BPP 类的 NP -集。结果在合理的假设条件下, 零知识证明可用于证明任意 NP -集的成员问题。直观上, 可用一个 NP -完全集来阐明这一事实, 以下重点讨论 3-着色图集合的零知识证明系统。

要证明一个给定图 G 是 3-着色图, 只需给出 G 的一种 3-着色方案即可 (同样可证明 NP 中任意集合的成员问题), 但是这种 NP -证明并非零知识证明 (除非 $NP \subseteq BPP$)。事实上, 假设 $NP \not\subseteq BPP$, 则图的 3-着色问题不存在零知识 NP -证明系统。然而, 我们将看到, 图的 3-着色问题确实具有交互式零知识证明系统。描述该证明系统时, 使用一些隐藏信息, 随后再揭示这些信息的 “盒子”。这些盒子可用单向函数 (见定理 9.11) 实现。

构造 9.10 (3-着色性的零知识证明, 抽象描述) 构造中使用一些不透明的盒子, 可以妥善地锁上或打开, 且上锁之后能很好地隐藏信息。

- 公共输入。一个简单图 $G = (V, E)$ 。

① 事实上, 这与构造 9.3 的合理性分析中的现象相同。

② 这种观察并非关键, 但是确实可以简化分析 (去掉了每次调用验证者策略时, 要规定一个随机序列的需求)。

③ 回顾一个概率称为不可忽略的, 如果它大于某个正多项式 (关于相关参数) 的倒数。

• 证明者的第一步。设 ψ 为 G 的一种 3-着色方案。证明者随机选择集合 $\{1, 2, 3\}$ 上的一种置换 π ，并对每个 $v \in V$ ，令 $\phi(v) \stackrel{\text{def}}{=} \pi(\psi(v))$ 。从而，证明者对 3-着色方案 ψ 重新进行了随机标记。证明者传给验证者 $|V|$ 个不透明且上锁的盒子，使得第 v 个盒子中包含 $\phi(v)$ 。

• 验证者的第一步。验证者均匀选择一条边 $\{u, v\} \in E$ ，发送给证明者。

• 动机说明。这些盒子中应该包含图的一种 3-着色方案，而验证者发出询问来检查顶点 u 和 v 的颜色。事实上，在零知识条件下，规定证明者只对相应于图中边的顶点对的询问作出回答是非常关键的。

• 证明者的第二步。收到边 $\{u, v\} \in E$ 后，证明者发给验证者对应于 u 和 v 的盒子的钥匙。

为简化分析，如果验证者发送了 $\{u, v\} \notin E$ ，则假设证明者的行为与收到 E 中的（随机）边一样，而不是挂起交互过程，这是很正常的。

• 验证者的第二步。验证者打开对应于 u 和 v 的盒子，当且仅当其中包含 $\{1, 2, 3\}$ 中两个不同元素时，接受证明。

构造 9.10 中的验证者策略易在概率多项式时间内实现。如果给定图 G 的一种 3-着色方案作为辅助输入，则证明者策略也易在概率多项式时间内实现。显然，如果输入的图是 3-着色图，则验证者在与证明者交互之后，接受的概率为 1。另一方面，如果输入图不是 3-着色图，则放入盒子中的任何内容至少对一条边而言是无效的，于是，验证者会以至少 $1/|E|$ 的概率拒绝。因此，上述协议对 3-着色图与非 3-着色图的接受概率之间存在一个不可忽略的缝隙。为增加该缝隙，必须重复协议足够多次（当然，每次重复中使用独立的随机数）。

到此之止，我们已经证明构造 9.10 构成了图的 3-着色性的（一种较弱形式的）交互式证明系统。然而，关键在于预定的证明者策略必须为零知识的。在构造 9.10 的抽象环境中，证明者策略的零知识性易证，因为验证者在真正交互中看到的所有信息是一些盒子序列以及一对随机的不同颜色（这是易模仿的）。事实上，在对真正交互的模拟过程中，需要提供一系列的盒子和一对随机的不同颜色，作为验证者指定的两个盒子的内容。注意：上述讨论依赖于一个事实，即（由验证者指定的）盒子对应于图中有边相连的顶点。

对零知识性的这种简单解释是无法数字化实现的，因为盒子并非完全不受其内容影响（实际上是以一种不可区分的方式被其内容影响）。因此，验证者可能会根据盒子的“外在特征”选择要检查的边，而盒子的“外在特征”又依赖于（以不可区分的方式）其内容。结果选择了一对盒子之后，根本无法判断其内容，所以上述的简单模拟无法实现。相反，我们用如下的交互方法来模拟。

(1) 首先（随机地）猜测试验者要打开的一对盒子（对应于一条边），并在这两个盒子中随机放入两种不同颜色（而在其他盒子中放入无用数据）^①。然后将所有盒子送给验证者，验证者再要求打开（相应于一条边的）一对盒子。

(2) 如果验证者所要求的一对盒子正是一开始选择的（即猜测成功时），则打开这些盒子，完成模拟过程；否则，再次尝试（即重复第一步，提出新的随机猜测和随机颜色）。关键在于如果从盒子的外表无法区分盒中内容，则猜测成功的概率近似为 $1/|E|$ 。进一步地，这个模拟在执行时要与真正的交互无法区分。

因此，使用几乎完全（而非完全不透明）隐藏其内容的盒子就足够了。这种盒子可用数

① 另一种（更高效的）模拟中，将随机独立的颜色放入不同盒子，并希望验证者请求得到一个正确着色的边。在假设盒子中内容能（几乎）完全保密时，这一事件发生的概率（约）为 $2/3$ 。

字方法实现。

教学注释：建议在课堂上只讲述并分析上述的抽象协议，并简单提及数字化实现，不必对定理 9.11 作严格的证明（可见参考文献[100]（或参考文献[91]中第 4 节））。

数字化实现（概述）

可以使用充分定义的承诺方案来实现抽象的盒子（见构造 9.10）。

不严格地，该方案是在发送方和接收方之间进行的一个两阶段游戏，第一阶段结束后，发送方“承诺”了一个值，而在该阶段，接收方不可能得到这个值（即承诺是“隐藏的”）。在第二阶段，发送方向接收方揭示承诺的值（即承诺是“绑定的”）。假设单向函数存在，则这种承诺方案可以实现（如定义 7.3），见附录 C 的 C.4.3.1 节。

对其他 NP-集的零知识证明

利用图的 3-着色性为 NP-完全问题这一事实，可以得到（由构造 9.10）任意 NP-集的零知识证明系统。^①进一步地，NP-证据可以高效地转化为实现相应（预定为零知识的）证明者策略的多项式规模电路。

定理 9.11（ZK 定理） 假设（非均匀困难的）单向函数存在，则有 $NP \subseteq ZK$ 。进一步地，假设给定关于 S 中公共输入成员判定的 NP-证据作为辅助输入，则任意 $S \in NP$ 都具有概率多项式时间内实现的证明者策略。

定理 9.11 中的假设（即单向函数的存在性）似乎无可避免，因为对于“平均情况下困难”的问题存在着零知识证明，这蕴含了单向函数的存在性（并且类似地，BPP 类之外的集合存在零知识证明，蕴含了“辅助输入的单向函数”的存在性）。

定理 9.11 对于密码协议的设计有极其深远的影响（见附录 C）。我们指出，在同样假设下，任意交互式证明可以转化为零知识证明（然而，转化中不一定保持证明者的复杂度）。

定理 9.12（终极 ZK 定理） 假设存在（非一致困难的）单向函数，则 $IP = ZK$ 。

不严格地说，定理 9.12 可利用 $IP = AM(\text{poly})$ 证明，并可将任意公开掷硬币协议做如下修改：让证明者发送关于其消息的承诺而非消息本身，交互结束后，证明者将（零知识地）证明相应的消息副本被原始验证者接受。事实上，后一个断言属于“NP 类型”，并因此可以调用定理 9.11 中的零知识证明系统来证明。

思考

定理 9.11 的证明利用了 3-着色性为 NP-完全问题这一事实，为了得到 NP 中任意集合的零知识证明而利用了 3-着色性协议（即构造 9.10）。因此，这里是以“正面”方式使用 NP-完全性结论，即为了构造某个东西，而不是得到某种困难性（“负面”）结论（见 2.2.4 节）。^②

完全及统计零知识性

上述结论针对的是计算上的零知识证明系统，可以将其与统计零知识证明系统的已知复杂性结论进行比较：统计零知识证明只对 $IP(2) \cap \text{co}IP(2)$ 中的集合存在，而不太可能对所有 NP-集都存在。另一方面，已知的统计零知识类中包含了某些困难问题，并被证明具有有趣的复杂性理论性质（如取补运算下的封闭性，以及具有自然的完全问题）。感兴趣的读者可见参考文献[227]。

① 实际上，应该或者依赖于标准 Karp-归约在多项式时间内不可逆，或者依赖于 3-着色协议关于辅助输入（如定义 C.9）是零知识的。

② 历史上，定理 9.11 的证明可能是 NP-完全性的第一次正面应用。后来对完全性结论的正面应用还出现于交互式证明（见定理 9.4 的证明）、概率可检验证明（见定理 9.16 的证明）及统计零知识的研究中（见参考文献[227]）。

9.2.3 知识的证明——附加内容

教学注释：技术上，这个内容应属于 9.1 节，但其中更有趣的是涉及到了知识的零知识证明——因此，把它单独作为一个小节。

笼统地说，“知识的证明”是这样一种交互式证明，其中证明者宣布拥有关于某对象（如某个图的 3-着色方案）的“知识”，而不是仅仅知道其存在性（如该图存在着 3-着色方案，这又等价于宣布该图为 3-着色图）。注意：宣布知识的实体实际上就是证明者策略，它是一种自动计算设备，以下用机器来指代。这就引出一个问题，称“一个机器知道某事”究竟是什么意思？

9.2.3.1 抽象的思考

任何一部词典中都给出了动词“知道”的几种含义，但这些解释都参考了“意识”的概念，这一概念显然无法用于计算机中。因此，我们应该寻求动词“知道”的行为主义解释。事实上，将知识与做某事的能力（如能写下知道的东西）相关联是合理的。从而，如果一个机器可以输出字符 α ，可以说它“知道” α 。但这样做似乎也没有意义：机器的输出是有良好定义的——输出或者等于 α ，或者不等于，所以，称一个机器“能做到某事”，这又是什么意思呢？

有趣的是，对这个问题的合理解释尚不存在。大体上，称“一个机器能做到某事”，是指该机器易被修改为（或其修改后的版本）能做到其宣称的事。更准确地，这意味着存在一种高效机器，使用原先的机器作为黑盒（或给定其代码作为输入）时，可以输出其宣称的任何信息。

技术上讲，使用交互式机器（即可以实现交互式策略）作为黑盒似乎更有意义。这就是我们讨论的核心内容。进一步地，从概念上讲，无论机器知道（或不知道）什么都是其自身的事，而我们感兴趣的是通过与该机器交互可以推断出何种知识。因此，真正有意义的是对知识的证明（而非知识本身）。

9.2.3.2 一个具体例子

为简单起见，考虑一个具体问题：如何才能证明一个机器知道某个图的 3-着色方案？显然，可将该着色方案直接发送给验证者。但我们认为，构造 9.10 中的协议（即 3-着色性的零知识证明系统）是证明图的 3-着色性的另一种方法。

3-着色方案的验证者定义中涉及到可能的证明者策略，并将从证明者处“提取”（对给定图的）一种 3-着色方案的可能性与证明者使验证者确信的的概率相关联。也就是说，定义中假定存在一种高效的通用方法，当证明者策略使验证者以概率 1（或更一般地，以某个显著概率）接受给定图时，该方法能根据证明者策略来“提取”一种 3-着色方案。另一方面，我们不应期望该提取器能从无法使验证者确信（或更一般地，以可忽略的概率使其确信）的证明者策略那里得到任何信息。更健壮的定义应允许两种极端情况间（具体而言，在使验证者以显著概率确信的证明者与使其以可忽略概率确信的证明者之间）的平滑转换。还应该认为 Alice 的下述策略在直觉上为零知识的：假设 Bob 已经成功地使 Alice 确信他知道给定图的一种 3-着色方案，则 Alice 发送给 Bob 该着色方案。^①我们强调这里 Alice 策略的零知识性应该与用于证明 Bob 知道 3-着色方案的知识证明系统无关。

^① 为简单起见，读者可以考虑具有唯一 3-着色方案的图（在对图中顶点重新标号的情况下）。一般是指具有唯一解的实例（见 6.2.3 节），这在许多（密码学）应用中是很自然的。

不严格地，称一个策略 V 构成了 3-着色知识的验证者，如果对任意证明者策略 P ，将 P 当作“黑盒”^①而提取图 G 的一种 3-着色方案的复杂度与 P 使 V 确信（接受图 G ）的概率成反比。也就是说，由称为提取器的预言机完成对 3-着色方案的提取，该提取器可以访问策略 P （即规定 P 对于其收到信息如何应答的函数）。要求提取器在输入 G 和预言访问 P 之下的（期望）运行时间与 P 使 V 确信接受 G 的概率成反比（因子为 $|G|$ 的多项式）。特别地，如果 P 总是使 V 接受 G ，则提取器运行于期望的多项式时间内。同样，在 P 使 V 以显著概率接受时也是如此。另一方面，如果 P 永远不会使 V 接受，则对提取器不作任何要求。我们强调后一种特殊情况不满足定义，见参考文献[91]中 4.7.1 节的讨论。

知识的证明，特别是知识的零知识证明，在密码方案及协议的设计中有许多应用（见参考文献[91, 92]）。这由下述的通用结论保证。

定理 9.13（定理 9.11，重述）假设（非一致困难的）单向函数存在，则任意 NP-关系具备知识的零知识证明（对于相应的 NP-证据）。进一步地，在给定这样一个 NP-证据时，预定的证明策略可在概率多项式时间内实现。

9.3 概率可检验证明系统

教学注释：概率可检验证明（PCP）系统可以看作是一种受限类型的交互式证明系统，其中证明者是无记忆的，而对每个验证者消息应答时都将该消息看成是第一个消息。从这个角度可以与前两小节内容建立更紧密的联系，但是稍显牵强。事实上，这种无记忆的证明者可以被视为一种静态对象，验证者可以任选位置发出询问。但在描述这种模型时，使用（更传统的）预言机术语而非（退化的）交互机（或策略），是更吸引人的作法。

概率可检验证明系统在标准（确定型）证明系统上附加了一个概率程序，能通过检查证明中几个位置来评估断言的正确性。本节将重点放在这种概率程序上，因为其中蕴含了确定型验证程序（遍历概率程序所有可能的随机选择，并作出充分测试）的存在性。

这种概率检验程序可以测试证明中的某些位置，故对其建立模型时要求程序能直接访问证明中的单个比特（从而不需要逐比特地扫描证明）。因此，可将要检验的证明看作一个串，这与传统证明系统相同，但是（概率）检验程序可以直接访问该串中的单个比特（图 9.3）。

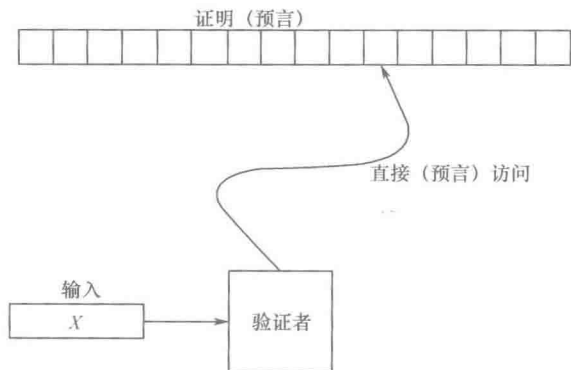


图 9.3 PCP 模型——图示

① 事实上，还可以考虑无黑盒子的提取器。

我们感兴趣的是, 概率检验程序只访问证明中几个位置, 但仍可对证明的正确性作出有意义的概率判定。具体而言, 检验程序应该(以概率 1) 接受任意正确证明, 但是以至少 $1/2$ 的概率拒绝错误证明。这种概率检验程序称为概率可检验的证明 (PCP) 系统。

可以通过检查证明中个别位置来(有意义地) 评估证明正确性, 这一事实令人惊奇且与直觉相悖。当然, 这种证明必须以非标准方式写出, 因为标准的证明如果不全部读出, 是无法检验其正确性的(因为一个不恰当的推理就可能导致错误)。相反, PCP 系统的证明似乎非常充分, 它们包含了过多(关于要证明断言) 的信息片段, 通过检查随机选择的几个相关片段的一致性便可(有意义地) 评估证明的正确性。我们强调, 这里“有意义的评估”是指以常数概率(而不是与证明长度成反比的概率) 拒绝错误断言。

PCP 系统的主要复杂性度量是其询问复杂度。另一个自然的度量是证明的长度, 这又与系统的随机性复杂度相关。PCP 的随机性复杂度在许多应用(如 PCP 系统的构成以及应用 PCP 来得到不可近似性结论) 中起着重要作用, 因此我们将更详细地讨论这个参数。

教学注释: 事实上, PCP 系统最著名的应用是证明大量的不可近似性结论(见 9.3.3 节), 但我们认为后者仅仅是 PCP 系统基本概念的一个重要应用, 据此来组织本节内容。

9.3.1 定义

大体上, 概率可检验证明系统包含一个概率多项式时间验证者, 它可以访问表示证明的预言机(以冗余形式)。验证者只访问少数几个预言比特, 这些比特的位置由验证者随机选择。与交互式证明系统相同, 要求如果断言成立, 则验证者总是接受(即当给定充分的预言访问时); 然而, 如果断言错误, 则无论是否使用预言, 验证者必须以至少 $1/2$ 的概率拒绝。PCP 环境的基本定义在以下定义的第一部分给出。然而, 第二部分引入的复杂性量度对于后续讨论具有重要意义。

定义 9.14 (概率可检验证明——PCP)

(1) 集合 S 的一个概率可检验证明系统 (PCP) 是一个概率多项式时间预言机 V , 称为验证者, 满足以下两个条件。

- 完备性。对任意 $x \in S$, 存在一个预言 π_x , 使得对输入 x 以及 π_x 的预言访问, 机器 V 总是接受 x 。
- 合理性。对任意 $x \notin S$ 以及任意预言 π , 对于输入 x 及预言 π 的访问, 机器 V 以至少 $1/2$ 的概率拒绝 x 。

(2) 称一个概率可检验证明系统具有询问复杂度 $q: \mathbf{N} \rightarrow \mathbf{N}$, 如果对任意长为 n 的输入, 验证者至多发出 $q(n)$ 个预言询问。^①类似地, 随机性复杂度 $r: \mathbf{N} \rightarrow \mathbf{N}$ 为验证者对于 n 比特输入需要产生的随机数数量上限。对于整数函数 r 和 q , 将具有随机性复杂度 r 和询问复杂度 q 的概率可检验证明系统集合记作 $PCP(r, q)$ 。对于整数函数集合 R 和 Q , 有

$$PCP(R, Q) \stackrel{\text{def}}{=} \bigcup_{r \in R, q \in Q} PCP(r, q)$$

多次重复应用证明系统, 可以减小 PCP 系统的误差概率(在合理性条件中)。特别地, 重复证明过程 k 次后, 可将验证者被错误断言欺骗的概率降低到 2^{-k} , 而复杂度至多增加了一

^① 按照复杂性理论中的习惯, 预言应答为二进制(即 0 或 1)。

个因子 k 。因此, 具有非平凡询问复杂度 (见 9.3.2 节) 的 PCP 系统在证明中被测试的位置数量与对断言正确性的确信程度之间达成折中。

要注意, PCP 系统完备性条件中的预言 π_x 构成了标准数学意义上的证明。事实上, 从任意一个 PCP 系统都可得到一个标准证明系统 (验证程序扫描 V 可能生成的所有随机数, 并模拟所有相应的检查)。进一步地, PCP 系统中具有对数随机性复杂度的预言构成了 NP-证明 (见习题 9.15)。然而, 这些预言还有另一个值得注意的性质, 那就是允许一个“懒”验证者生成随机数, 然后碰碰运气, 在不读完证明全过程 (而是只读出很小一部分) 的情况下“评估”证明的正确性。这就潜在地允许验证者只检查一个 NP-证明中很少几个比特, 甚至可以使用极长的证明 (即超多项式长度)。

适应性与非适应性的验证者

定义 9.14 允许验证者为适应性的, 即验证者可以根据之前的询问应答 (除了根据输入和验证者内部生成随机数之外) 来确定其询问。相反, 非适应性的验证者只根据输入和内部生成的随机数确定询问。注意: q 个适应性的 (二进制) 询问可以用 $\sum_{i=1}^q 2^{i-1} < 2^q$ 个非适应性的 (二进制) 询问来模拟。我们指出, 大多数 PCP 系统在构造时都使用了非适应性的验证者, 且许多资料中的 PCP 系统都定义为非适应性的。

随机性与证明长度

给定验证者 V , 称 (预言中的) 位置 i 与输入 x 相关, 如果存在 V 对输入 x 的计算, 其中对位置 i 提出询问 (即存在 ω 和 π , 使得对输入 x , 随机数 ω , 以及对预言 π 的访问, 验证者询问位置 i)。 V 的有效证明长度是指满足对任意输入 x , (预言中) 最多有 $l(|x|)$ 个位置与 x 相关的最小函数 $l: \mathbf{N} \rightarrow \mathbf{N}$ 。我们断言, 任意 PCP 系统的有效证明长度与随机性 (及询问) 复杂度紧密相关。另一方面, 如果 PCP 系统具有随机性复杂度 r 和询问复杂度 q , 则其有效证明长度的上限为 2^{r+q} , 而在非适应性系统中, 上限为 $2^r \cdot q$ (见习题 9.15)。因此, 具有对数随机性复杂度的 PCP 系统其有效证明长度是多项式的, 由此得到 NP-证明系统。另一方面, 在某种意义上, PCP 系统的随机性复杂度上限可以是证明长度的对数 (如果允许非一致的验证者, 见习题 9.16)。

随机性的作用

PCP 定理 (即 $\mathcal{NP} = \mathcal{PCP}(\log, O(1))$) 称可以根据常数个测试比特对证明进行有意义的评估。要注意的是, 当要求验证者为确定型时, 除非 $\mathcal{P} = \mathcal{NP}$, 否则, 这是不可能的。首先, 注意到 $\mathcal{PCP}(0, O(1)) = \mathcal{P}$ (作为 $\mathcal{PCP}(r, q) \subseteq D_{\text{TIME}}(2^{2^{r+q}} \cdot \text{poly})$ 的一种特殊情况, 见习题 9.17)。其次, 如习题 9.19, $\mathcal{P} \neq \mathcal{NP}$ 中蕴含了 $\mathcal{NP}$ 不包含于 $\mathcal{PCP}(o(\log), o(\log))$ 中。最后, 假设并非所有的 NP-集都具有 NP-证明系统, 其中使用的证明长度为 l (如 $l(n) = n$), 则当 $2^{r(n)} q(n) < l(n)$ 时, $\mathcal{PCP}$ 不包含于 $\mathcal{NP}$ 中 (见习题 9.17)。

9.3.2 概率可检验证明的功能

著名的 PCP 定理认为, $\mathcal{NP} = \mathcal{PCP}(\log, O(1))$, 这个结论是本小节的核心。但在讨论它之前, 要简单介绍一下 PCP 层级。

首先注意 $\mathcal{PCP}(\text{poly}, 0)$ 等价于 coRP , 而 $\mathcal{PCP}(0, \text{poly})$ 等价于 $\mathcal{NP}$ 。易证明, PCP 层级

中集合的非确定时间复杂度上限的一个结论（见习题 9.17）。

命题 9.15（PCP 的功能上限） 对任意多项式界整数函数 r ，有 $PCP(r, \text{poly}) \subseteq N_{\text{TIME}}(2^r \cdot \text{poly})$ 。特别地， $PCP(\log, \text{poly}) \subseteq \mathcal{NP}$ 。

重点讨论对数随机性复杂度的 PCP 系统，体现了对使用多项式长度预言机的 PCP 系统的兴趣（见 9.3.1 中的讨论）。我们强调，这种 PCP 系统（即 $PCP(\log, q)$ ）是一些 NP-证明系统，且具有一个（令人惊奇的）附加性质：可以通过测试证明的一（小）部分（即 $q(n)$ 比特）来“概率地评估”断言的正确性。因此，对任意固定的多项式界函数 q ，形如

$$\mathcal{NP} \subseteq PCP(\log, q) \quad (9.6)$$

的结论是有趣的（因为它还可应用于证据长度超过 q 的 NP-集）。当然， q 越小越好。PCP 定理给出一个令人惊奇的事实： q 可以为常量。

定理 9.16（PCP 定理） $\mathcal{NP} \subseteq PCP(\log, O(1))$ 。

因此， $\mathcal{NP}$ 中每个集合都存在这样的概率可检验证明，其中验证者只须生成对数个随机数并只进行常数次的询问。该常数基本上就是 3（见 9.3.4.1 节）。在证明定理 9.16 之前，我们先给出几点说明。

NP-证据到 PCP 预言的高效转换

定理 9.16 的证明是构造性的，其中可将（ $\mathcal{NP}$ 中集合的一个实例的）任意 NP-证据高效地转化为使 PCP 验证者（以概率 1）接受的预言。也就是说，对每个（NP-证据关系） $R \in \text{PC}$ ，存在一个如定理 9.16 中的 PCP 验证者 V ，以及多项式时间计算的函数 π ，使得对任意 $(x, y) \in R$ ，当给定对证明 $\pi(x, y)$ 的预言访问时，验证者 V 总是接受输入 x （即 $\Pr[V^{\pi(x, y)}(x) = 1] = 1$ ）。回顾这里的预言访问本身也是 NP-证明，从而 NP-证明可以转化为能在读出证明比例与确信程度之间提供折衷的 NP-证明。具体地，对任意 $\varepsilon > 0$ ，如果能容忍错误概率 ε ，则只需测试（转化后的）NP-证明中 $O(\log(1/\varepsilon))$ 个比特即可。实际上，需要随机选择这些比特位置（如 9.3.1 节中所讨论的）。

上述对定理 9.16 的强化扩展了定理的应用范围。定理 9.16 本身足以提供“负面”应用，如证明某些近似问题的不可解性（见 9.3.3 节）。但是对于“正面”应用（见 9.3.4.2 节），典型地，某些用户（或一个真正的实体）会被要求构造出 PCP-预言，此时强化的定理 9.16 将会起作用。

NP 的一个特征：将定理 9.16 与命题 9.15 相结合，可以得到以下关于 NP 的一个特征。

推论 9.17（NP 的 PCP 特征） $\mathcal{NP} = PCP(\log, O(1))$ 。

PCP 定理证明导引

定理 9.16 是一系列出色工作中的巅峰，这些工作都提出并证明了有意义且逐渐增强版本的式 (9.6)。定理 9.16 的完整证明超出了本书范围（并且我们认为不适合在复杂性理论的基础教程中介绍）。因此，以下只给出原始证明（见 9.3.2.2 节）概述以及另一种证明方法（见 9.3.2.3 节），后者是在原始证明提出十多年后发现的。首先给出两种证明都用到的一个较弱结论，该结论（见 9.3.2.1 节）本身也有价值，其中称任意 NP-集具有常数询问复杂度的 PCP 系统（虽然随机性复杂度为多项式），即 $\mathcal{NP} \subseteq PCP(\text{poly}, O(1))$ 。

教学注释：我们认为，PCP 定理证明的任意一部分都不应作为课堂讲授内容。特别是应该先介绍 PCP 与近似复杂度之间的联系。在以下 3 个小节中，建议先讲 9.3.2.1 节，因为它直接阐述了 PCP 的功能。对 PCP 定理本身的两个证明，则要视具体情况。一方面，如果仅仅是为了介绍证明思想，则简单讲述原始证明（见 9.3.2.2 节）。另一方面，如果是为了给出一个完整的证明，则我们极力建议讲述新的证明方法（9.3.2.3 节中只给出概述）。

9.3.2.1 证明 $NP \subseteq PCP(\text{poly}, O(1))$

任意 NP-集具有常数询问复杂度的 PCP 系统（不考虑随机性复杂度），这已经证实了 PCP 系统的功能。其中称通过检查（可能很长的）证明中极少数位置就能检验证明的正确性。以下介绍的 PCP 系统使用指数长度的证明，但是只需检查证明中的常数个（随机选择的）位置。

先简述这种构造。首先要注意，对一个给定的 $GF(2)$ 上二次方程组，只需构造出一个证明其可满足性的 PCP 即可，因为该问题是 NP-完全问题（见习题 2.25）。^①PCP 的输入为含 n 个变量的二次方程组，(PCP 的) 预言要能计算所有这些 n 变量的二次表达式在某种给定赋值下的取值。假设该赋值能满足输入的二次方程组。在预言中区分两个表格：第一张表对应于所有 2^n 个线性表达式，第二张表对应于 2^{n^2} 个二次表达式。每张表都需（通过“线性测试”来）测试自包含性，且需测试两张表是否一致（利用“矩阵等价”测试，其中使用了“自纠错”）。最后，测试被这些表编码的赋值是否满足输入的二次方程组。方法是任取二次方程组中方程的一种随机线性组合，并利用上述表格得到对相应二次表达式的赋值（仍使用自纠错）。关键在于上述的每种测试都使用常数个布尔查询，且时间（及随机性）复杂度为输入长度的多项式。具体细节如下。

教学注释：以下内容涉及一些概念，如 Hadamard 编码、测试、自纠错等，本书其他部分（如附录 E 中 E.1.2.2、10.1.2 节以及 7.2.1.1 节）介绍了这些概念。本节内容是自包含的，当然更全面的介绍（在上述部分中）会更有用。

起点

我们构造 $GF(2)$ 上可满足的二次方程集合的 PCP 系统。输入为关于变量 $x_1, x_2, \dots, x_n$ 的一系列方程，证明预言包含两部分（或两张表），假设其中包含一些可满足的赋值 $\tau = \tau_1 \tau_2 \dots \tau_n$ （也可看作是 $GF(2)$ 上的 n 元向量）。第一部分，记作 T_1 ，对可满足赋值进行 Hadamard 编码，对任意 $\alpha \in GF(2)^n$ ，这张表中包含 n 元向量 α 和 τ 的内积 mod 2（即 $T_1(\alpha)$ 等于 $\sum_{i=1}^n \alpha_i \tau_i$ ）。第二部分，记作 T_2 ，包含了所有 $\tau_i \tau_j$ 的线性组合，即对任意 $\beta \in GF(2)^{n^2}$ （可看作 $GF(2)$ 上的 $n \times n$ 矩阵）， $T_2(\beta)$ 等于 $\sum_{i,j} \beta_{i,j} \tau_i \tau_j$ （事实上， T_1 包含于 T_2 中，因为对任意 $\sigma \in GF(2)$ ，有 $\sigma^2 = \sigma$ ）。PCP 验证者利用两张表检查输入（即一系列二次方程）是否可被表中编码的赋值满足。当然，表中可能包含一些内容，并非任意 n 元向量的有效编码（更不要说满足输入方程），所以验证者还需检查编码是否正确。先解决这个任务。

^① 本书用 $GF(2)$ 表示含 2 个元素的域。

Hadamard 编码的测试

注意：假设 T_1 对线性函数编码，即一定存在某些 $\tau = \tau_1 \tau_2 \cdots \tau_n \in \text{GF}(2)^n$ ，使得对任意 $\alpha = \alpha_1 \cdots \alpha_n \in \text{GF}(2)^n$ ，有 $T_1(\alpha) = \sum_{i=1}^n \tau_i \alpha_i$ 。为测试这个假设，可以均匀选择 $\alpha', \alpha'' \in \text{GF}(2)^n$ ，并检查是否有 $T_1(\alpha') + T_1(\alpha'') = T_1(\alpha' + \alpha'')$ ，其中 $\alpha' + \alpha''$ 表示 $\text{GF}(2)$ 上的向量加法。对这种测试的分析相当复杂，而实际情况是任何一个与线性表距离为 0.02 的表将至少以 0.01 的概率被拒绝（见习题 9.20），称 T 与线性表的距离为 ε ，是指 T 与任意线性函数 f 在定义域内不同的比例至多为 ε （即 $\Pr[T(r) \neq f(r)] > \varepsilon$ ）。

通过常数地重复线性测试，可以以至少 0.99 的概率拒绝与 Hadamard 码距离为 0.02 的表。所以，利用常数次询问，验证者可以拒绝与任意 $\tau \in \text{GF}(2)^n$ 的 Hadamard 码距离为 0.02 的 T_1 。类似地，拒绝与任意 $\tau' \in \text{GF}(2)^{n^2}$ 的 Hadamard 码距离为 0.02 的 T_2 。因此，可以假设 T_1 （或 T_2 ）与某个 τ （或 τ' ）的 Hadamard 码距离为 0.02^①（当然，这并不意味着 τ' 等于 τ 与自身的外积（即 $\tau'_{i,j}$ 不一定等于 $\tau_i \tau_j$ ））。

在剩余的分析中，固定 $\tau \in \text{GF}(2)^n$ 和 $\tau' \in \text{GF}(2)^{n^2}$ ，并将 τ （或 τ' ）的 Hadamard 码记作 $f_\tau: \text{GF}(2)^n \rightarrow \text{GF}(2)$ （或 $f_{\tau'}: \text{GF}(2)^{n^2} \rightarrow \text{GF}(2)$ ）。同时提醒大家注意 T_1 （或 T_2 ）与 f_τ （或 $f_{\tau'}$ ）为 0.02-接近的。

Hadamard 码的自纠错性

假设 T 与线性函数 $f: \text{GF}(2)^m \rightarrow \text{GF}(2)$ 距离为 ε （即 $\Pr_r[T(r) \neq f(r)] \leq \varepsilon$ ），则可以在任意点 x ，通过对 T 的两次（随机）询问来恢复 f 的值。具体地，对均匀选择的 $r \in \text{GF}(2)^m$ ，利用 $T(x+r) - T(r)$ 。注意：能恢复正确值的概率至少为 $1 - 2\varepsilon$ ，因为 $\Pr_r[T(x+r) - T(r) = f(x+r) - f(r)] \geq 1 - 2\varepsilon$ 且由 f 的线性性，有 $f(x+r) - f(r) = f(x)$ （当然，当 $\varepsilon < 1/4$ 时，函数 T 不可能与两个不同线性函数均为 ε -接近）。^②因此，假设 T_1 与 f_τ 为 0.02-接近（或 T_2 与 $f_{\tau'}$ 为 0.02-接近），则可以通过在任意点发出两次向 T_1 （或 T_2 ）的询问而正确的恢复（即以 0.04 的错误概率） f_τ （或 $f_{\tau'}$ ）值。这个过程称为自纠错（见 7.2.1.1 节）。

检查 f_τ 与 $f_{\tau'}$ 的一致性

假设给定了对 $f_\tau: \text{GF}(2)^n \rightarrow \text{GF}(2)$ 和 $f_{\tau'}: \text{GF}(2)^{n^2} \rightarrow \text{GF}(2)$ 的访问，其中 $f_\tau(\alpha) = \sum_i \tau_i \alpha_i$ 且 $f_{\tau'}(\alpha') = \sum_{i,j} \tau'_{i,j} \alpha'_{i,j}$ ，我们希望能验证对任意 $i, j \in \{1, 2, \dots, n\}$ ，是否有 $\tau'_{i,j} = \tau_i \tau_j$ 。换言之，在给定对两个矩阵 $A = (\tau_i \tau_j)_{i,j}$ 和 $A' = (\tau'_{i,j})_{i,j}$ 的编码时，希望能检查这两个矩阵是否相同。可以证明（见习题 9.22），如果 $A \neq A'$ ，则 $\Pr_{r,s}[r^T A s \neq r^T A' s] \geq 1/4$ ，其中 r 和 s 为均匀分布的 n 元向量。注意：在这种情况下（即 $A = (\tau_i \tau_j)_{i,j}$ 和 $A' = (\tau'_{i,j})_{i,j}$ ），有 $r^T A s = \sum_j \left(\sum_i r_i \tau_i \tau_j \right) s_j = f_\tau(r) f_\tau(s)$ （图 9.4）且 $r^T A' s = \sum_j \left(\sum_i r_i \tau'_{i,j} \right) s_j = f_{\tau'}(rs^T)$ ，其中 rs^T 是 s 与 r 的外积。因此，（对于

① 注意： τ （或 τ' ）由 T_1 （或 T_2 ）唯一确定，因为任意两个不同的 $\text{GF}(2)^m \rightarrow \text{GF}(2)$ 的线性函数恰好在定义域的一半取值相同（即 Hadamard 码的相对距离为 1/2）。

② 事实上，这一事实可由自纠错性的讨论中得到，但是一个更简单的证明仅涉及 Hadamard 码的相对距离为 1/2 这一事实。

$(\tau_i \tau_j)_{i,j} \neq (\tau'_{i,j})_{i,j}$ 有 $\Pr_{r,s} [f_r(r) f_r(s) \neq f_{r'}(rs^T)] \geq 1/4$ 。

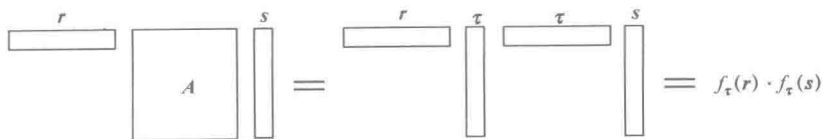


图 9.4 线性及二次方程的一致性测试

然而，注意到，我们并不能直接访问函数 f_r 和 $f_{r'}$ ，只能访问与这些函数 0.02-接近的表格（即 T_1 和 T_2 ）。但是利用自纠错，可以以很高的概率得到 f_r 和 $f_{r'}$ 在任意点的值。实际上，在实现上述的一致性测试时，只利用 f_r 的自纠错就够了，因为使用了 f_r 在 $\text{GF}(2)^n$ 上两个独立均匀分布的点的取值（即 r 和 s ），而只使用了 $f_{r'}$ 在 rs^T 的值，但 rs^T 在 $\text{GF}(2)^{n^2}$ 中并非均匀分布。因此，通过均匀选择的 $r, s \in \text{GF}(2)^n$ 和 $R \in \text{GF}(2)^{n^2}$ ，并检查是否有 $T_1(r)T_1(s) = T_2(rs^T + R) - T_2(R)$ ，可以测试 f_r 与 $f_{r'}$ 的一致性。

常数重复上述（自纠错）一致性测试之后，能以至少 0.99 的概率拒绝不一致的表。所以在剩余分析中，假设 $(\tau_i \tau_j)_{i,j} = (\tau'_{i,j})_{i,j}$ 。

检查 τ 是否满足二次方程组

假设给定对上述 f_r 和 $f_{r'}$ 的访问（特别地， $\tau' = \tau\tau^T$ ）。如果 τ 不满足（二次）方程组，则也将以 1/2 的概率不满足这些方程的任意线性组合。为了检查 τ 是否满足（作为输入的）二次方程组，取这些方程的一种随机线性组合来构造一个二次方程，并检查该方程是否可被 τ 满足。关键在于对 τ 是否满足二次方程 $Q(x) = \sigma$ 的测试实际上等价于测试是否 $f_{r'}(Q) = \sigma$ 。在实际测试中，仍可利用（表 T_2 的）自纠错。

这就完成了对验证者的描述。注意到，该验证者对 Hadamard 编码实施了常数次数测试，以及常数次一致性和可满足性测试，而在每次可满足性测试中都使用了 Hadamard 编码的自纠错。每次测试使用常数次询问（2~4 次），且使用的随机数个数变量个数的平方（以及输入方程个数的线性函数）。因此，询问复杂度是一个常量，而随机性复杂度最多为输入（二次方程组）长度的平方。显然，如果输入的二次方程组是可满足的（被某个 τ ），则验证者以概率 1 接受相应的表 T_1 和 T_2 （即 $T_1 = f_r$ ， $T_2 = f_{\tau\tau^T}$ ）。另一方面，如果输入的二次方程组是不可满足的，则任意一对表 (T_1, T_2) 将以常数概率被拒绝（在上述的一次测试中），从而有 $\mathcal{NP} \subseteq \mathcal{PCP}(q, O(1))$ ，其中 q 为二次多项式。

思考

事实上，在二次方程组可满足性的实际测试中，由于可满足的赋值（在预言中）以冗余方式编码，且符合最终的可满足性测试，从而可使测试更便捷。但此时测试的负担转移到了检查这种编码是否正确上。实际上，由上述验证者进行的大多数测试其目标是验证编码的正确性。这种（对编码的）正确性测试可以看作是测试编码各部分间的一致性。所有这些问题都可在 PCP 系统更高级的构造中找到。

9.3.2.2 PCP 定理的第一种证明概述

PCP 定理（定理 9.16）的原始证明主要包括以下 3 个步骤，后面将进一步讨论这些步骤。

第一步：对具有对数随机性复杂度和多项式—对数询问复杂度的 $\mathcal{NP}$ 集，构造一个（非适应的）PCP 系统，即该 PCP 具有期望的随机性复杂度和非常低（但不为常数）的询问复杂度。另外，该证明系统还有其他一些性质，可以帮助实现以下第三步中的证明组合。

第二步：对具有多项式随机性复杂度和常数询问复杂度的 $\mathcal{NP}$ 集构造一个 PCP 系统，即该 PCP 具有期望的（常数）询问复杂度，但是随机性复杂度高得离谱（事实上，9.3.2.1 节中已经给出了这样一种构造）。另外，该证明系统也具有能实现第三步中证明组合的附加性质。

第三步：证明组合模型。^①一般来说，模型将两个证明系统组合，使得可以用“内部”验证者概率地验证“外部”验证者的接受标准。即组合后的验证者为“外部”验证者选择随机数，确定“外部”验证者要检查的（证明中）相应位置，并验证“外部”验证者应该已经接受了这些位置处的值。后一个验证过程通过调用“内部”验证者来实现，并无需读出上述位置处的值。这样做的目标是引导（“组合后的”）验证，并使询问数量远小于“外部”证明系统的询问复杂度。特别地，内部验证者无法读出其输入，这使得组合过程比术语本身更巧妙。

大体上，外部验证者应该是强壮的，其合理性条件应以较高概率保证预言应答“远离”满足剩余的判定谓词（而不仅仅是不满足）（进一步地，这里所说的谓词由外部验证者的非适应性定义，且其中必有一个规模上限为询问数量多项式的回路）。内部验证者可以访问输入，并应答每个询问，但只要求其（以很高概率）拒绝远离正确的输入（并且照例接受正确的输入）。也就是说，内部验证者实际上是一个近似验证者。

将两个这样的 PCP 组合起来可以得到关于 $\mathcal{NP}$ 的一个新的 PCP，其中新的证明预言由“外部”系统的证明预言和一系列“内部”系统的证明预言（为“外部”验证者每个可能的随机带提供一个“内部”证明）组成。最终得到的验证者为外部验证者选择随机数并利用相应的“内部”证明来验证外部验证者是否接受所选择的随机数。注意这时选择的随机数确定了外部验证者将检查的“外部”证明中的位置，而组合后的验证者为内部验证者提供了对这些位置的预言访问（这被内部验证者当成输入）以及相应“内部”证明的预言访问（这被内部验证者当作证明预言），如图 9.5 所示（后面将提供进一步的细节）。

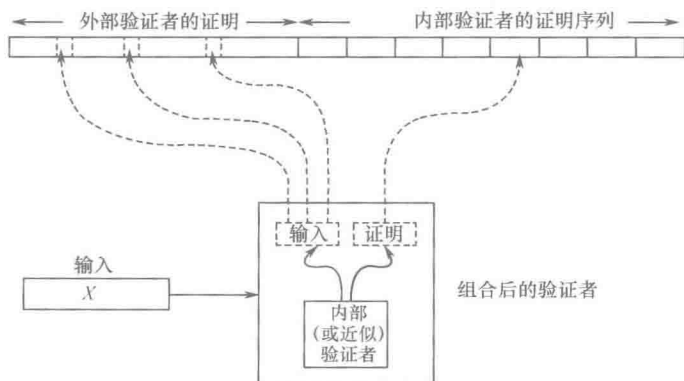


图 9.5 PCP 系统的组合——图示。箭头由内部验证者的（虚拟）输入及证明预言指向组合后验证者的实际证明。这些指针（及剩余谓词）通过调用外部验证者来确定

注意：将一个随机性复杂度为 r' 、询问复杂度为 q' 的外部验证者与随机性复杂度为 r'' 、询问复杂度为 q'' 的内部验证者相结合，可得到一个新的 PCP，其随机性复杂度为

^① 我们采用了参考文献[35] 中的组合模型，而没有用参考文献[15, 16]中的原始版本。

$r(n) = r'(n) + r''(q'(n))$ ，而询问复杂度 $q(n) = q''(q'(n))$ ，这是由于 $q'(n)$ 表示内部验证者访问的输入（预言）长度。注意：外部验证者是非适应的，从而如果内部验证者也是非适应的（或健壮的），则组合得到的验证者是非适应的，当需要将这个验证者与其他内部验证者相组合时，这一点很重要。

特别地，将第一步中的证明系统与其自身相组合（利用 $r'(n) = r''(n) = O(\log n)$ 及 $q'(n) = q''(n) = \text{poly}(\log n)$ ），可得到一个（关于 $\mathcal{NP}$ 的）PCP 系统，其随机性复杂度为 $r(n) = r'(n) + r''(q'(n)) = O(\log n)$ ，询问复杂度为 $q(n) = q''(q'(n)) = \text{poly}(\log \log n)$ 。将后者（作为“外部”系统）与第二步中的 PCP 系统组合，可得到（ $\mathcal{NP}$ 的）一个 PCP 系统，随机性复杂度为 $r(n) + \text{poly}(q(n)) = O(\log n)$ ，询问复杂度为 $O(1)$ ，这样就证明了 PCP 定理。

一个更详细的概述——具体构造

上面的描述利用两个（非平凡的）PCP 系统，并利用了一些附加性质如健壮性以及近似验证。9.3.2.1 节中已经给出了一个具有多项式随机性复杂度和常数询问复杂度的 PCP 系统（如第二步中所假设的）。所以这里先讨论近似验证和健壮性的概念，并阐述其在该 PCP 中的应用。然后，详细介绍“外部”健壮的 PCP 与“内部”近似 PCP 的组合。最后，概述其他一些常用的 PCP 系统（即在第一步所假设的）。

近似 PCP

在标准 PCP 中，验证者得到一个明确输入，并可对证明过程进行预言访问（关于输入是否属于某个预定集合成员）。相反，近似 PCP 中，验证者可以（直接）访问两个预言，一个代表输入，另一个为相应证明，对两个预言的询问都计入其询问复杂度中。典型地，该验证者的询问复杂度小于输入预言长度，所以验证者不能读出全部输入，也不能期望其了解全部输入。事实上，验证者不会去判定输入是否属于某个集合，而只要求其区分输入属于某集合以及输入与该集合距离较远这两种情况（距离较远的含义是相对汉明距离至少为 0.01（或其他较小常数））。

例如，考虑 9.3.2.1 节中系统的一种变形，其中二次方程组是固定的^①，验证者要确定在输入预言中出现的赋值是否满足给定方程组，或与任意使方程组满足的赋值距离较远。我们使用如 9.3.2.1 节中的证明预言，以及如 9.3.2.1 节中的近似验证者，另外，验证者还要实施近似性测试来检查输入是否接近证明预言中的赋值编码。具体地，验证者在输入预言中读取均匀选择的比特，并与由证明预言中得到的自纠错值相比较（即对均匀选择的 $i \in \{1, 2, \dots, n\}$ ，比较输入预言中第 i 比特与证明预言中得到的 $T_1(0^{i-1}10^{n-i})$ 的自纠错值）。

健壮的 PCP

如果外部验证者能以“健壮”的方式拒绝，则“外部”PCP 验证者与“内部”PCP 的近似验证者的组合才有意义。因此，在健壮验证者的合理性条件中，要求（至少以 1/2 的概率）预言应答与被剩余谓词接受的任意序列距离较远（而不是预言应答仅仅被该谓词拒绝）。即对任意“否”实例 x ，及每个证明 $\pi = \pi_1 \pi_2 \dots \pi_l \in \{0, 1\}^l$ ，要求对于验证者选择的随机数 $\omega \in \{0, 1\}^r$ ， $\pi_{i_{\omega,1}} \pi_{i_{\omega,2}} \dots \pi_{i_{\omega,q}}$ 至少以 1/2 的概率远离任意满足 P_ω 的赋值，其中 $i_{\omega,j}$ 为对 ω 的第 j 个（非适应性）询问，而 P_ω 为确定此时何种应答序列可被接受的剩余谓词。事实上，如果外部验证者是

① 在这里的应用中，二次方程组是被（“内部”）验证者“已知的”，因为它由（“外部”）验证者确定。

健壮的，则它可以区分正确应答与远离正确赋值的应答。

例如，如果健壮性是用相对常数距离来定义的（实际情况如此），则 9.3.2.1 节中的 PCP（以及任意具有常数询问复杂度的 PCP）一定是健壮的。然而，我们不考虑这种情况，因为在证明组合中只把这种 PCP 作为内部验证者。相反，我们会考虑作为外部验证者的 PCP 的健壮性（如第一步中假设的 PCP，以下介绍）。

对证明组合的进一步说明

在上面的概述之后，我们进一步讨论本节使用的证明组合。先详细介绍两种要组合的 PCP。设 V_1 为一个健壮的验证者，具有随机性复杂度 r_1 和询问复杂度 q_1 ，并假设其对输入 x 和随机带 $\omega \in \{0,1\}^{\eta(|x|)}$ 的剩余判定可用规模为 $\text{poly}(q_1(|x|))$ 的电路来描述，该电路记作 C_ω 。即对于输入 x ，给定对预言 $\pi = \pi_1\pi_2 \cdots \pi_l$ 和随机带 $\omega \in \{0,1\}^{\eta(|x|)}$ 的访问，当且仅当 $C_\omega(\pi_{i_{\omega,1}}\pi_{i_{\omega,2}} \cdots \pi_{i_{\omega,q_1(|x|)}}) = 1$ 时，验证者 V_1 接受 x ，其中 $i_{\omega,j}$ 是对输入 x 和随机带 ω 进行的第 j 次询问。注意到， $C_\omega^{-1}(1)$ 的成员可以在时间 $\text{poly}(|C_\omega^{-1}|) = \text{poly}(q_1(|x|))$ 内确定。令 V_2 为关于 $C_\omega^{-1}(1)$ 成员问题的近似验证者，并假设其近似参数等于（或小于） V_1 的健壮性参数。实际上，验证者 V_2 应该或者依赖于电路 C_ω ，或者得到 C_ω 的描述作为辅助输入。^①现在转向组合后得到的验证者，首先假设，对于输入 x ，该验证者利用形如 $\left(\pi, \left(\pi^{(\omega)}\right)_{\omega \in \{0,1\}^{\eta(|x|)}}\right)$ 的证明，其中 π 为 V_1 的证明（关于输入 x ）， $\pi^{(\omega)}$ 为 V_2 的证明（关于串 $\pi_{i_{\omega,1}}\pi_{i_{\omega,2}} \cdots \pi_{i_{\omega,q_1(|x|)}}$ 是否为 $C_\omega^{-1}(1)$ 的成员）。组合后得到的验证者（为 V_1 ）均匀选择一个随机带 $\omega \in \{0,1\}^{\eta(|x|)}$ ，确定位置 $i_{\omega,1}, i_{\omega,2}, \dots, i_{\omega,q_1(|x|)}$ （ V_1 将在这些位置对输入 x 和随机带 ω 进行询问），并在假设其可以访问输入预言 $\pi_{i_{\omega,1}}\pi_{i_{\omega,2}} \cdots \pi_{i_{\omega,q_1(|x|)}}$ 和证明预言 $\pi^{(\omega)}$ 时，调用 V_2 。即如果 V_2 询问其输入中（或证明中）第 j 个比特，则组合后的验证者询问 π 的第 $i_{\omega,j}$ 个比特（或 $\pi^{(\omega)}$ 的第 j 个比特），并向 V_2 提供得到的比特。

显然，如果 x 是一个“是”实例，则利用充分证明 π 和 $\left(\pi^{(\omega)}\right)_{\omega \in \{0,1\}^{\eta(|x|)}}$ ，可使组合后的验证者以概率 1 接受。另一方面，如果 x 是一个“否”实例，则 V_1 将以至少 1/2 的概率“健壮地拒绝”任意 π （即针对选择 $\omega \in \{0,1\}^{\eta(|x|)}$ ， $\pi_{i_{\omega,1}}\pi_{i_{\omega,2}} \cdots \pi_{i_{\omega,q_1(|x|)}}$ 与集合 $C_\omega^{-1}(1)$ 中任意串距离都较远的概率至少为 1/2）。现在，如果 V_1 在使用随机带 $\omega \in \{0,1\}^{\eta(|x|)}$ 时，能“健壮地拒绝” π ，则（对任意 $\pi^{(\omega)}$ ） V_2 在相应执行中的拒绝概率至少为 1/2。从而对任意选择的证明预言（即任意 π 及 $\left(\pi^{(\omega)}\right)_{\omega \in \{0,1\}^{\eta(|x|)}}$ ），则组合后的验证者以至少 1/4 的概率拒绝每个“否”实例。当然，可以通过串行重复来增加拒绝概率。

① 在前一种情况中， V_2 是一个电路（可以访问其输入和证明预言），其中集成了电路 C_ω 。在后一种情况中，近似 PCP 的定义应该扩展到包含了输入被分成两部分的情况，其中第一部分（如 C_ω ）是明确给定的（作为通常的输入），而第二部分（如 C_ω 的输入）通过预言访问来间接给定。无论何种方法，要求 C_ω 的规模为其自身输入长度的多项式（即 $|C_\omega| = \text{poly}(q_1(|x|))$ ），这很关键。事实上，一种渐近的处理方法是利用后一种定义（有两部分输入）。此时， V_2 实际上是对 $P \subseteq NP$ 中断言的一个（扩展的）近似 PCP，而正确断言具有形式 (C, α) 使得 $C(\alpha) = 1$ （其中 C 为明确的输入，而 α 为间接输入）。

教学注释:很遗憾,构造具有对数随机性复杂度和多项式—对数询问复杂度的(NP集的)PCP需要许多技术细节。另外,如何获得这种PCP的健壮版本也超出了本章范围。因此,以下描述仅仅提供了构造思路。

NP集的具有对数随机性复杂度和多项式—对数询问复杂度的PCP

主要证明 $NP \subseteq PCP(f, f)$, 其中 $f(n) = \text{poly}(\log n)$, 该结论只需对原始证明稍作修改(在后面讨论)即可得到。对 $NP \subseteq PCP(f, f)$ 的证明系统建立在对3CNF范式算术化的基础上,这与9.1.3.2节中使用的不同(其中为coNP构造一个交互式证明系统)。首先先描述这种算术化,再介绍建立于其上的PCP系统。

在算术化过程中,3CNF范式 ϕ 的变量(或子句)名称表示为 $(|\phi|)$ 的对数长度的二进制串,而 ϕ 中普通变量(或子句)用对数个新的变量代替,并为这些新变量分配有限域 $F \supset \{0, 1\}$ 中的值。剩余讨论中的运算均指有限域 F (含 $\text{poly}(|\phi|)$ 个元素)中的运算。3CNF范式 $\phi(x_1, x_2, \dots, x_n)$ (的结构)可表示为布尔函数 $C_\phi: \{0, 1\}^{O(\log n)} \rightarrow \{0, 1\}$, 满足 $C_\phi(\alpha, \beta_1, \beta_2, \beta_3) = 1$ 当且仅当 $i = 1, 2, 3$ 时, ϕ 的第 α 个子句中的第 i 个文字具有索引 $\beta_i = (\gamma_i, \sigma_i)$, 这可看作是变量名加符号。因此,对任意 $\alpha \in \{0, 1\}^{\log |\phi|}$, 存在唯一的 $(\beta_1, \beta_2, \beta_3) \in \{0, 1\}^{3 \log 2n}$, 使得 $C_\phi(\alpha, \beta_1, \beta_2, \beta_3) = 1$ 。下面考虑 F 上对 C_ϕ 的一种多线性扩展,记作 Φ , 即 Φ 为 $\{0, 1\}^{O(\log n)} \subset F^{O(\log n)}$ 上与 C_ϕ 一致的(唯一的)多线性多项式。

现在转到PCP, 首先要注意,验证者可将最初的3SAT-实例 ϕ 简化为上述的算术实例 Φ , 即对于输入的3CNF范式 ϕ , 验证者首先构造 C_ϕ 和 Φ (如习题7.12)。验证者的一部分证明预言被当作函数 $A: F^{\log n} \rightarrow F$, 它被看作是满足 ϕ 的真值赋值的一种多线性扩展(即对任意 $\gamma \in \{0, 1\}^{\log n} \equiv [n]$, $A(\gamma)$ 的值为这种赋值中第 γ 个变量的值)。因此,我们希望检查对任意 $\alpha \in \{0, 1\}^{\log |\phi|}$, 是否有

$$\sum_{\beta_1, \beta_2, \beta_3 \in \{0, 1\}^{3 \log 2n}} \Phi(\alpha, \beta_1, \beta_2, \beta_3) \cdot \prod_{i=1}^3 (1 - A'(\beta_i)) = 0 \quad (9.7)$$

式中: $A'(\beta)$ 为(变量)赋值 A 下第 β 个文字的值,即对 $\beta = (\gamma, \sigma)$, 其中 $\gamma \in \{0, 1\}^{\log n}$ 为变量名而 $\sigma \in \{0, 1\}$ 为文字类型(即变量是否取反),有 $A'(\beta) = (1 - \sigma) \cdot A(\gamma) + \sigma \cdot (1 - A(\gamma))$ 。因此,式(9.7)成立当且仅当由 A 诱导的赋值使第 α 个子句满足(因为该子句的三个文字 β 中,至少有一个使 $A'(\beta) = 1$ 一定成立)。^①

与9.3.2.1节相同,我们无法验证式(9.7)的所有 $|\phi|$ 个实例。进一步地,不同于9.3.2.1节,我们也无法取 $|\phi|$ 个实例的一种随机线性组合(因为这需要太多的随机数)。幸运的是,利用这些等式的“伪随机”线性组合也是可以的。具体来说,可以利用足够的(可高效构造的)小偏移概率空间(见8.5.2.3节)。将这种空间(规模为 $\text{poly}(|\phi| \cdot |F|)$)而偏移至多为 $1/6$ 记作 $S \subset F^{|\phi|}$, 可以均匀选择 $(s_1, s_2, \dots, s_{|\phi|}) \in S$ 并检查是否有

^① 注意:对这样的 α , 存在唯一的三元组 $(\beta_1, \beta_2, \beta_3) \in \{0, 1\}^{3 \log 2n}$ 使得 $\Phi(\alpha, \beta_1, \beta_2, \beta_3) \neq 0$ 。这个三元组 $(\beta_1, \beta_2, \beta_3)$ 为出现在第 α 个子句中的文字编码,该子句被 A 满足当且仅当存在 $i \in [3]$ 使得 $A'(\beta_i) = 1$ 。

$$\sum_{\alpha\beta_1\beta_2\beta_3\in\{0,1\}^l} s_\alpha \cdot \Phi(\alpha, \beta_1, \beta_2, \beta_3) \cdot \prod_{i=1}^3 (1 - A'(\beta_i)) = 0 \quad (9.8)$$

式中: $l \stackrel{\text{def}}{=} \log|\phi| + 3\log 2n$ 。小偏移性质保证了如果 A 无法满足任意一个形如式 (9.7) 的方程, 则 A 无法满足式 (9.8) 的概率至少为 $1/3$ (对所有选择的 $(s_1, s_2, \dots, s_{|\phi|}) \in S$)。由于 $|S| = \text{poly}(|\phi| \cdot |F|)$, 而非 $|S| = 2^{|\phi|}$, 我们可以利用 $O(\log|\phi|)$ 个随机数在 S 中选择一个样本。这样就将原始问题简化为对随机选择的 $(s_1, s_2, \dots, s_{|\phi|}) \in S$, 检查式 (9.8) 是否成立。

(暂且) 假设 A 为低次多项式, 可以利用“和测试”(与 coNP 的交互式证明类似) 来概率地验证式 (9.8), 即在迭代中分割 l 个二进制和。每次迭代中, 验证者得到相应的单变量多项式, 并在随机点为其赋值。事实上, 验证者可以通过足够多的询问 (其中规定了和测试中所有选择的序列) 来得到相应的单变量多项式。^① 注意: 在对 l 个和进行分割之后, 验证者最后提出一个表达式, 其中包含 A' 中的 3 个未知变量, 这可通过向 A 提出相应询问来获取。和测试需要选择 $l \cdot \log|F|$ 个随机数, 并发出 $(l+3) \cdot O(\log|F|)$ 次布尔询问 (这对应于 l 次询问, 每次的应答为 (F 上的) 一个常数次多项式), 以及 3 个向 A 的询问 (应答为 F 中元素)。令 $|F| \gg O(l)$, 其中 $l = O(\log|\phi|)$, 可证明和测试的合理性。

然而, 假设 A 为低次多变量多项式是不恰当的。相反, 必须检查 A 是否真的是低次多变量多项式 (或接近于这样的多项式), 并利用自纠错来恢复 A 的值 (这在上述和测试中是必需的)。幸运的是, 与这些和测试具有类似复杂度的“低次测试”^② 确实存在 (并且可在该复杂度内实现自纠错)。因此, 利用有限域 F 中的 $\text{poly}(\log(n))$ 个元素, 可得到 $\text{NP} \subseteq \text{PCP}(f, f)$, 其中 $f(n) \stackrel{\text{def}}{=} O(\log(n)) \cdot \log \log(n)$ 。

为了得到具有对数随机性复杂度的 PCP 系统, 用 $\{1, 2, \dots, \log n\}$ 上长为 $\frac{O(\log n)}{\log \log n}$ 的序列重新命名最初的变量和子句, 而不是使用对数长度的二元序列。这要求使用低次多项式扩展 (即次数为 $(\log n) - 1$ 的多项式), 而非多线性扩展。也可利用含 $\text{poly}(\log(n))$ 个元素的有限域, 于是, 在和测试及低次测试中只需要 $\frac{O(\log n)}{\log \log n} \cdot O(\log \log n)$ 个随机比特。然而, (为得到测试应答所需的) 询问的次数会增加, 因为目前使用的多项式次数为 $(\log n) - 1$ 而不是常数。这仅仅意味着询问复杂度增加了一个因子 $\frac{\log n}{\log \log n}$ (因为多项式次数增加了 $\log n$ 倍, 而变量个数减少了 $\log \log n$)。由此得到, $\text{NP} \subseteq \text{PCP}(\log, q)$, 其中 $q(n) \stackrel{\text{def}}{=} O(\log^2 n)$ 。

注意: 健壮性与 PCP 近似。

为了利用组合中的后一个 PCP, 需要确保该 PCP (或它的一个版本) 是健壮的, 并能提供一个近似的 PCP 版本。获得后一个版本相对比较容易 (利用 9.3.2.1 节中的思想), 而要满足健

① 询问中还包含序列 $(s_1, s_2, \dots, s_{|\phi|}) \in S$, (由验证者) 随机选择并在剩余步骤中固定下来。

② 这里所说的低次测试是一个预言机, 以概率 1 接受任意 (F 上) 的低次多项式, 并 (以至少 $1/2$ 的概率) 拒绝任意远离低次多项式的函数。参考文献 [195] 中提出了一种适当的测试 (还可见于习题 9.23)。

壮性则非常困难，此处省略。我们建议为了得到一个健壮的 PCP 系统，可以应用 PCP 系统（随机—高效的）“并行化”（见参考文献[15]），这又在很大程度上依赖于高效的低次测试。另一种方法（见参考文献[35]）利用了和测试的特殊结构（以及简单低次测试明显的健壮性。）

思考

PCP 定理称一个 PCP 系统可以同时达到最小的随机性复杂度和询问复杂度（在假设 $P \neq NP$ 时，可与原始值相差一个乘法因子）。上述构造通过组合两个不同的 PCP 而得到了这个出色的结论：第一个 PCP 达到了对数的随机性复杂度，但要使用多项式—对数个查询；第二个 PCP 使用常数次查询，但随机性复杂度为多项式。我们强调这两个 PCP 系统的非平凡性，且其自身也很有价值。还要强调一个事实，这些 PCP 可以利用非常简单的技术组合（这涉及到一些附加性质如健壮性和近似测试）。^①

9.3.2.3 PCP 定理的第二种证明概述

PCP 定理原始证明的重点在于构造两个 PCP 系统，它们是非平凡的且自身也有意义，然后将其以一种自然的方式组合。不严格地，这种组合（通过证明组合）保持了两个系统的良好性质，即可以得到一个 PCP 系统，继承了一个系统的（对数）随机性复杂度和另一个系统的（常数）询问复杂度。相反，以下要介绍的证明方法主要通过对数个步骤的渐近过程对 PCP 系统进行（性能上的）“放大”。我们从一个平凡的“PCP”系统出发，该系统具有期望的复杂度，但是拒绝错误断言的概率与其长度成反比，在每一步中，会使拒绝概率加倍，而仍保持原复杂度。即在每一步中，验证者的询问复杂度保持为常数，而随机性复杂度只增加一个常数项。这个过程渐进地将一个极弱的 PCP 系统转化为一个非常强的 PCP 系统（即如 PCP 定理中假设的系统）。

为了描述上述过程，需要重新定义 PCP 系统，使其允许任意合理的错误。出于技术上的考虑，将该过程描述为一个“受限的可满足性”问题到自身的迭代归纳将更方便。具体地，针对一个 2 变量约束系统，这可以用现成的（标记）图表示，使得图中结点对应于（非布尔的）变量，而边与约束条件相关联。

定义 9.18（具有 2-变量约束的 CSP） 对于给定有限集 Σ ，CSP 的实例包含一个图 $G=(V,E)$ ，（其中可能有并行边和环），以及与图中边相关联的 2-变量约束条件序列 $\Phi=(\phi_e)_{e \in E}$ ，每个约束条件具有形式 $\phi_e: \Sigma^2 \rightarrow \{0,1\}$ 。一个赋值 $\alpha: V \rightarrow \Sigma$ 的值是 α 满足的约束条件个数。即 α 的值为 $\left| \{(u,v) \in E: \phi_{(u,v)}(\alpha(u), \alpha(v)) = 1\} \right|$ 。把在最优赋值下不满足的约束条件所占比例记作 $\text{vlt}(G, \Phi)$ ，即

$$\text{vlt}(G, \Phi) = \min_{\alpha: V \rightarrow \Sigma} \left\{ \frac{\left| \{(u,v) \in E: \phi_{(u,v)}(\alpha(u), \alpha(v)) = 0\} \right|}{|E|} \right\} \quad (9.9)$$

对于各种函数 $\tau: \mathbb{N} \rightarrow (0,1]$ ，考虑承诺问题 $\text{gap CSP}_\tau^\Sigma$ ，其实例具有上述形式，且“是”实例为完全可满足的实例（即 $\text{vlt}=0$ ），而“否”实例为使 $\text{vlt}(G, \Phi) \geq \tau(|G|)$ 成立的对 (G, Φ) ，其中 $|G|$ 表示图 G 中边的个数。

^① 我们指出，PCP 系统的组合有可能缺少这些附加性质，但却更复杂。在某种意义上，这种新的组合需要将给定 PCP 系统转化成具有与健壮性和近似测试相关的性质。

注意: 对于 $\tau_0(m) = 1/m$, 3SAT 可归约为 $\text{gap CSP}_{\tau_0}^{\{1,2,\dots,7\}}$, 见习题 9.24。我们的目标是对某个固定的有限 Σ 及常数 $c > 0$, 将 3SAT (或 $\text{gap CSP}_{\tau_0}^{\{1,2,\dots,7\}}$) 归约为 $\text{gap CSP}_c^{\Sigma}$ 。如果能对 $\text{gap CSP}_c^{\Sigma}$ 给出一个简单的 PCP 系统, 则可得到 PCP 定理, 见习题 9.26 (9.3.3 节将进一步讨论约束可满足的问题与 PCP 定理间的关系)。期望的由 $\text{gap CSP}_{\tau_0}^{\Sigma}$ 向 $\text{gap CSP}_{\Omega(1)}^{\Sigma}$ 的归约可以通过对数次重复应用以下归约来得到。

引理 9.19 (gap CSP 到自身的放大归约) 对某个有限的 Σ 及常数 $c > 0$, 存在多项式时间计算的函数 f , 使得对 gap CSP^{Σ} 的每个实例 (G, Φ) , $(G', \Phi') = f(G, \Phi)$ 为 gap CSP^{Σ} 的一个实例, 且两个实例间的关系如下。

- (1) 如果 $\text{vlt}(G, \Phi) = 0$, 则 $\text{vlt}(G', \Phi') = 0$ 。
- (2) $\text{vlt}(G', \Phi') \geq \min(2 \cdot \text{vlt}(G, \Phi), c)$ 。
- (3) $|G'| = O(|G|)$ 。

也就是说, 可满足的实例被映射为可满足的实例, 而对不满足占比例 v 的约束条件的实例, 将其映射为不满足至少 $\min(2v, c)$ 比例的约束条件的实例。进一步地, 映射最多使 (实例中的) 边数增加一个常数因子。我们强调, Φ 和 Φ' 中都包含定义于 Σ^2 上的布尔约束条件。因此, 通过对数次重复应用引理 9.19, 可将 $\text{gap CSP}_{\tau_0}^{\Sigma}$ 归约为 $\text{gap CSP}_{\Omega(1)}^{\Sigma}$, 由此证明, $3\text{SAT} \in \text{PCP}(\log, O(1))$ (细节见习题 9.24 和习题 9.26)。

证明概要: ①在证明之前, 要强调证明中需解决的难点。具体地, 引理中断言“违背放大效应”(即第 (1) (2) 条), 在保持字母表 Σ 的同时, 只允许图规模的适度增长 (即第 (3) 条)。放弃后一个条件, 则可得到一个相对简单的证明, 其中模仿相应 PCP (一个扩张版本) ②的“并行重复”。因此, 证明中面临的挑战是要显著降低由并行重复引起的“规模扩张”, 并保持一个固定的字母表。第一个目标 (即第 (3) 项) 要求适当的去随机化, 而实际上应该使用扩张的随机漫游发生器 (见 8.5.3 节)。读过 9.3.2.2 节后, 读者可能会猜测第二个目标 (即固定的字母表) 可以利用证明组合模式解决 (我们在介绍本次证明时, 尽量让没有读过 8.5.3 节和 9.3.2.2 节的读者也能理解)。

为证明上述引理, 采用三步归约过程。第一步为预处理, 使实例中的图适于进一步分析 (如得到的图应该为一个扩张图)。在这一步中, vlt 值将减少一个常数因子。第二步是归约的核心, 其中为 vlt 值增加一个期望的常数因子。这需要构造当前图中常数长度的随机漫游。最后一步将使字母表 Σ 增大, 因此还需要一个后处理步骤来控制字母表的大小 (通过内部 PCP 系统, 如 9.3.2.1 节中所述)。具体细节如下。

首先要强调, 上述的 Σ 和 c , 以及辅助参数 d 和 t (后面将引入), 都是固定的常数, 这些参数的确定需要满足 (讨论中出现的) 各种条件。特别地, t 将是最后一个被确定的参数 (它比由其他所有参数共同确定的一个常数要大)。

从预处理步骤开始。这一步的目标是将输入的 gap CSP^{Σ} 的实例 (G, Φ) 归约为实例

① 细节可见参考文献[67]。

② 这里用扩张是为了避免使用参考文献[185]中的并行重复定理。在扩张版本中, 以常数概率 (如 1/2) 在包含相同变量 (或询问) 副本的元组间实施一致性检查。

(G_1, Φ_1) , 使得 G_1 为 d -正则图。^①进一步地, G_1 中每个顶点至少有 $d/2$ 个自环, 而边的数量保持在常数因子范围内 (即 $|G_1| = O(|G|)$), 且 $\text{vlt}(G_1, \Phi_1) = \Theta(\text{vlt}(G, \Phi))$ 。这一步很简单: 本质上是将最初的顶点用扩张图中度数成比例的顶点代替, 而将一个较大的扩张图添加到结果中 (见习题 9.27)。

主要步骤的目标是使违背约束条件的比例增加一个足够大的常数因子。直觉上, 这一步是为了使扩张图 G_1 上的随机漫游 (经过 t 条边) 与某个固定边集相交的概率接近于随机采样的 (t 条) 边和该集合相交的概率。因此, 可以期望漫游中经过违背条件的边的概率为 $\min(\Theta(t \cdot v), c)$, 其中 v 为这种边占有所有边的比例。事实上, 当前步骤是要将 gap CSP^Z 的实例 (G_1, Φ_1) 归约为 $\text{gap CSP}^{Z'}$ 的实例 (G_2, Φ_2) , 使得 $\Sigma' = \Sigma^{d'}$, 并满足以下条件。

(1) G_2 与 G_1 的顶点集完全相同, 且 G_1 中每个含 t 条边的路径由 G_2 中相应的边代替, 因此构成了一个 d' -正则图。

(2) Φ_2 中的约束条件将 Σ' 中每个元素当作对一个顶点的邻居 (距离 $\leq t$) 的 Σ -标记 (图 9.6), 并要求 (G_2 的边中端点的) 两个相应标记是一致的且满足条件 Φ_1 。即以下两种类型的条件可由 Φ_2 的约束条件来实现。

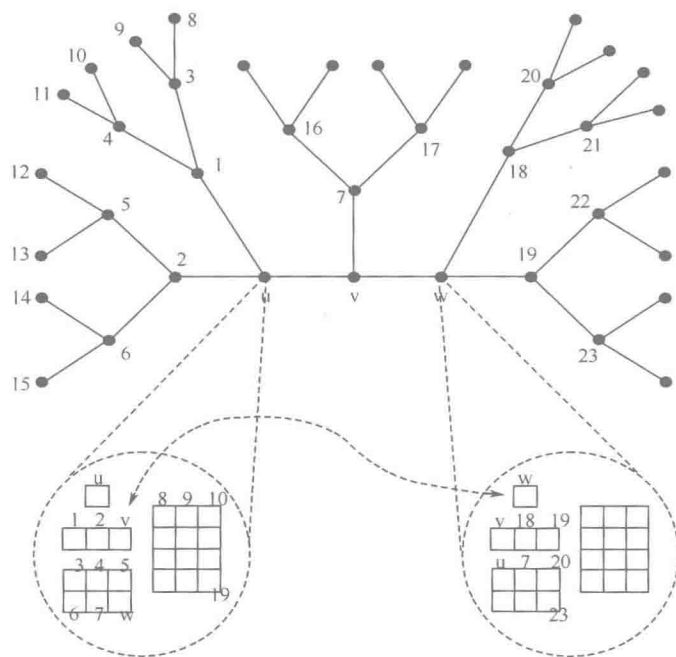


图 9.6 放大的归约。字母表 Σ' 是对距离 $t=3$ 的邻居的标记, 其中省略了重复。这里 $d=6$, 但没有画出自环 (从而“有效”度数为 3)。双向箭头表示 G_1 中的一条边影响了 (G_2, Φ_2) 中 u 与 w 间的约束

一致性: 如果顶点 u 和 w 在图 G_1 中由长度不超过 t 的路径相连, 且顶点 v 位于该路径中, 则与 G_2 中连接 u 与 w 的边相关联的 Φ_2 -约束条件要求顶点 u 和 w 在 Σ' -标记下对应于 v 的项

^① d -正则图是指每个顶点都恰好与 d 条边相关联的图。不严格地, 一个扩张图具有这样的性质, 每个适度平衡的分割 (即对图中顶点集进行划分) 具有足够多的交点。一个可用于实际分析的等价定义是, 除了最大特征值 (等于 d), 相应邻接矩阵的所有特征值其绝对值都与 d 相差较大。更多细节可见附录 E 中的 E.2.1.1 节。

相同。

满足 Φ_1 : 如果 G_1 中的边 (v, v') 位于由 u 出发, 长度不超过 t 的路径中, 则与 G_2 中对应于该路径的边相关联的 Φ_2 -约束条件实际上提供了与 (v, v') 关联的 Φ_1 -约束条件。

显然, $|G_2| = d^{t-1} \cdot |G_1| = O(|G_1|)$, 由于 d 为常量故 t 也将设为常量 (事实上, 适度增加图的规模, 可使在 G_1 中随机选择长为 t 的路径时, 选择过程复杂度较低)。

下面对这个步骤进行分析, 注意到, $\text{vlt}(G_1, \Phi_1) = 0$ 蕴含了 $\text{vlt}(G_2, \Phi_2) = 0$ 。一个有趣的事实是, 未满足的约束条件所占比例增加了一个因子 $\Omega(\sqrt{t})$, 即 $\text{vlt}(G_2, \Phi_2) \geq \min(\Omega(\sqrt{t} \cdot \text{vlt}(G_1, \Phi_1)), c)$ 。这里我们只陈述了一种直觉, 感兴趣的读者可以见参考文献[67]。主要考虑对 G_2 中顶点的 Σ' -标记中, 与 G_1 中某顶点的 Σ -标记相一致, 因为较少的不一致 (在由其他顶点的 Σ' -标记分配给某个顶点的 Σ -标记值中) 可以忽略, 而较多的不一致会使 G_2 中许多边的“相等条件”不满足。直观上, 利用 G_1 为扩张图的假设, 根据上述的 Σ -标记, (Φ_1 中) 被违背的边约束集将导致 Φ_2 中更多的边约束被违背 (因为 Φ_1 中每个边约束可由 Φ_2 中的许多边约束来实施)。要注意, G_1 中任意边集 F 出现于 G_2 中 (即 G_1 中长为 t 的路径) 的比例为 $\min(\Omega(t) \cdot |F|/|G_1|, \Omega(1))$ (注意: 当 G_1 为完全图时, 这一断言显然成立, 而对于扩张图, 结论也成立)。^①

第二步中得到的因子 $\Omega(\sqrt{t})$ 弥补了第一步中丢失的常数因子 (以及将在最后一步中丢失的常数因子)。进一步地, 对于恰当选择的常数 t , 上述增益是对 (vlt 的) 整体常数因子的放大。然而, 目前为止我们得到的只是 $\text{gap CSP}^{\Sigma'}$ 的实例, 而非 gap CSP^{Σ} 的实例, 其中 $\Sigma' = \Sigma^{d^t}$ 。最后一步的目的是将该实例归约为 gap CSP^{Σ} 实例。为此, 要将 $\text{gap CSP}^{\Sigma'}$ 的实例视为一个 (弱的) PCP 系统^② (类似于习题 9.26), 并利用 9.3.2.2 节中的证明组合模式将其与内部验证者相结合。我们强调这里的内部验证者只需要处理常数规模实例 (即描述长度为 $O(d' \log |\Sigma|)$), 于是, 9.3.2.1 节中的验证者即可满足要求。最终得到的 PCP 系统使用随机数 $r \stackrel{\text{def}}{=} \log_2 |G_2| + O(d' \log |\Sigma|)^2$, 以及常数二元询问, 其拒绝概率为 $\Omega(\text{vlt}(G_2, \Phi_2))$, 独立于 t 的选择。如习题 9.24, 对于 $\Sigma = \{0, 1\}^{O(1)}$, 易得到 gap CSP^{Σ} 的实例, 其不满足的约束比例为 $\Omega(\text{vlt}(G_2, \Phi_2))$ 。进一步地, 最终得到的实例 (作为三步归约的输出 (G', Φ')) 规模为 $O(2^r) = O(|G_2|)$, 这个等式利用了 d 与 t 为常数的事实。回顾 $\text{vlt}(G_2, \Phi_2) \geq \min(\Omega(\sqrt{t} \cdot \text{vlt}(G_1, \Phi_1)), c)$, 以及 $\text{vlt}(G_1, \Phi_1) = \Omega(\text{vlt}(G, \Phi))$, 引理得证。 $\square$

思考

9.3.2.2 节中的证明利用简单组合将两种构造相结合, 上述证明则与之相反, 它通过开发一种强大的“组合技术”来提高所使用的主要系统的性能。这种新方法称为放大引理 (引理 9.19), 它不仅能得到最优的组合系统, 而且还得到了比给定系统更好的系统。然而, 引理 9.19 所提供的性能放大相当温和, 为了得到期望结果, 需应用许多次。从另一方面看, 可以说好

① 由于技术上的困难, 证明界限 $\Omega(\sqrt{t} \cdot \text{vlt}(G_1, \Phi_1))$ 比证明 $\Omega(t \cdot \text{vlt}(G_1, \Phi_1))$ 要容易。

② 这里的 PCP-系统具有任意合理的误差 (即以概率 $\text{vlt}(G_2, \Phi_2) \in [0, 1]$ 拒绝实例 (G_2, Φ_2))。

的结果是通过许多温和的放大过程而渐进地得到的。

9.3.3 PCP 与近似

用概率可检验证明的术语描述 $\mathcal{NP}$ 类的特征,这在近似问题复杂性的研究中起着重要作用(见 10.1.1 节)。为阐述这种关系,首先要注意任意一个 PCP 系统 V 可以诱导出一个近似问题,其中要对给定输入估计最大接受概率,即对输入 x ,在给定对最优的 π 的预言访问下,求 V 接受 x 的近似概率(即求 $\max_{\pi} \{\Pr[V^{\pi}(x)=1]\}$ 的近似值)。因此,如果 $S \in \mathcal{PCP}(r, q)$,则判定 S 中成员的问题可归约为求 $\exp(2^{r+q})$ 个量的最大值(对应于所有高效预言),其中每个量可在时间 $2^r \cdot \text{poly}$ 内求值。这种归约(的正确性)至多为常数因子(即 2)的近似。

注意: 上述近似问题可表示为 PCP 验证者 V ,且其实例根据该验证者来赋值(即实例 x 的值为 $\max_{\pi} \{\Pr[V^{\pi}(x)=1]\}$)。本质上,这不是一个“自然的”近似问题。为了将 PCP 系统与自然的近似问题相关联,我们更深入地研究与 $\mathcal{PCP}(r, q)$ 相关的近似问题。

为简单起见,主要考虑非适应的 PCP 系统(即所有询问均基于输入和验证者内部生成的随机数而预先确定)。在这种系统中,固定输入 x ,对所有可能的内部随机数,验证者在检查证明预言中的相应比特之后进行判决,并用 $2^{r(|x|)}$ 个布尔公式表示这些判决。即 $2^{r(|x|)}$ 个公式中每一个都取决于表示证明预言中相应比特的 $q(|x|)$ 个布尔变量。因此,如果 x 为一个“是”实例,则(对这些变量)存在一种真值赋值,满足所有 $2^{r(|x|)}$ 个公式,而当 x 为“否”实例时,不存在一种真值赋值能满足多于 $2^{r(|x|)-1}$ 个公式。进一步地,当 $r(n) = O(\log n)$ 时,给定 x ,可在多项式时间内构造出相应的布尔公式序列。因此,PCP 定理(即定理 9.16)能得到对可同时满足的常数规模布尔公式进行近似计数的困难性结论。这就引出下述定义。

定义 9.20 (SAT 和推广 SAT 的缝隙问题) 对于常数 $q \in \mathbb{N}$ 和 $\varepsilon > 0$, 承诺问题 $\text{gapGSAT}_{\varepsilon}^q$ 的实例为 q -变量布尔公式序列(即每个公式至多依赖于 q 个变量)。其中“是”实例为可同时满足的公式序列,而“否”实例为不满足占序列中多于 $1-\varepsilon$ 比例公式的任意布尔赋值。承诺问题 $\text{gapSAT}_{\varepsilon}^q$ 也用类似方法定义,只有一点不同,其中每个实例为分离的子句序列(即序列中每个公式由单个子句构成)。

事实上, $\text{gapSAT}_{\varepsilon}^q$ 的每个实例可以自然地看作是 q -CNF 范式,而我们考虑的是满足尽可能多(输入 CNF 中)子句的赋值。根据提示, $\mathcal{NP} \subseteq \mathcal{PCP}(\log, O(1))$ 蕴含了 $\text{gapGSAT}_{1/2}^{O(1)}$ 的 NP-完全性,这又暗示了对某个常数 $\varepsilon > 0$, $\text{gapSAT}_{\varepsilon}^3$ 是 NP-完全问题,反之亦然。所有断言将在下面给出具体描述和证明。

定理 9.21 (PCP 定理的等价表示) 以下 3 个条件等价:

- (1) PCP 定理: 存在常数 q , 使得, $\mathcal{NP} \subseteq \mathcal{PCP}(\log, O(1))$ 。
- (2) 存在常数 q , 使得 $\text{gapGSAT}_{1/2}^q$ 为 $\mathcal{NP}$ -完全问题。
- (3) 存在常数 $\varepsilon > 0$, 使得 $\text{gapSAT}_{\varepsilon}^3$ 为 $\mathcal{NP}$ -完全问题。

定理 9.21 的要点不是其正确性(根据 3 个条件的正确性易证),而是证明的简单性。注意:在条件(2)和(3)中没有提到 PCP。因此,其与条件(1)的等价性(这是易证明的)表明对自然优化问题的近似是 PCP 定理的核心。一般而言, $\mathcal{NP}$ 问题的概率可检验证明系统

可以推出各种经典优化问题的强不可近似性结论（见习题 9.18 和 10.1.1 节）。

证明：先证 PCP 定理中蕴含了 gapGSAT 的 NP-困难性。不失一般性，可以假设对某个常数 q 和任意 $S \in \mathcal{NP}$ ，利用非适应的验证者（因为 q 次适应性的询问可以用 2^q 次非适应性询问来模仿），可以证明 $S \in \text{PCP}(O(\log), q)$ 。我们用如下方法将 S 归约为 gapGSAT。对于输入 x ，扫描验证者内部生成的所有可能的 $2^{O(\log|x|)}$ 个随机数序列，并对每个序列，确定验证者的询问以及剩余判定谓词（该谓词确定了对哪些序列的应答可以让验证者接受）。即对每个随机输出 $\omega \in \{0,1\}^{O(\log|x|)}$ ，考虑由 x 和 ω 确定的剩余谓词，其中详细规定了对于随机数 ω ，哪些 q -比特预言应答可以让验证者接受 x 。事实上，该谓词只依赖于 q 个变量（代表 q 个相应的预言应答）。将 x 映射为由 $\text{poly}(|x|)$ 个公式组成的序列，每个公式含 q 个变量，从而得到 gapGSAT q 的一个实例。这个映射可在多项式时间内计算，并且事实上将 $x \in S$ （或 $x \notin S$ ）映射为 gapGSAT $^q_{1/2}$ 的“是”实例（或“否”实例）。

要证明条件（2）蕴含着条件（3），可以构造 GSAT 到 3SAT 的标准归约。具体地，将 gapGSAT $^q_{1/2}$ 归约为 gapSAT $^q_{2^{-(q+1)}}$ ，当 $\varepsilon = 2^{-(q+1)}/(q-2)$ 时，这又可归约为 gapSAT $^3_\varepsilon$ 。注意：条件（3）蕴含了条件（2）（如给定 gapSAT $^3_\varepsilon$ 的一个实例，考虑在给定实例中，所有可能的 $1/\varepsilon$ 个分离子句的合取）。

最后证明条件（3）蕴含条件（1）（同样可证明条件（2）蕴含着条件（1））。主要思路是证明 gapSAT $^3_\varepsilon$ 属于 $\text{PCP}(\varepsilon^{-1} \log, 3\varepsilon^{-1})$ ，并利用 $\mathcal{NP}$ 到 gapSAT $^3_\varepsilon$ 的归约来得到 $\mathcal{NP}$ 中每个集合的 PCP。实际上，可以构造一个自然的 PCP 系统来证明 gapGSAT $^q_\varepsilon$ 属于 $\text{PCP}(\varepsilon^{-1} \log, \varepsilon^{-1} q)$ 。在这个 PCP 系统中，证明预言被认为是一种可满足性赋值，验证者在输入序列中随机选择一个（ q 变量）公式，并检查该公式是否能被预言（给出的赋值）满足。这可归结为选择对数个随机数并发出 q 次询问。验证者总是接受“是”实例（在给定充分预言时），而每个“否”实例被拒绝的概率至少为 ε （无论是否使用预言）。为了放大拒绝概率（到期望阈值，即 $1/2$ ），可以调用上述验证者 ε^{-1} 次（并注意 $(1-\varepsilon)^{1/\varepsilon} < 1/2$ ）。■

缝隙放大归约——思考

定理 9.21 中的条件（2）（或条件（3））蕴含了 GSAT（或 3SAT）可以归约为 gapGSAT $_{1/2}$ （或 gapSAT $^3_\varepsilon$ ）。这意味着，在一些问题（如 3SAT）中，存在到自身的“缝隙放大”归约，这些归约将“是”实例映射为“是”实例（按照惯例），而将“否”实例映射为与“是”实例距离较远的“否”实例。也就是说，“否”实例被映射为一种特殊类型的“否”实例，使得在“是”实例与作为映射的像的“否”实例间形成一个“缝隙”。例如，对于 3SAT 问题，不可满足的公式被映射为不但不可满足，而且还不存在一种赋值能使占比例超过 $1-\varepsilon$ 的子句满足的公式。因此，PCP 的构造本质上是“缝隙放大”归约。

9.3.4 对 PCP 自身的更多讨论：概述

首先讨论 NP 类的 PCP 特征变形，再讨论具有超出 NP 的特殊功能的 PCP。当然，后一种系统具有超对数（Superlogarithmic）的随机性复杂度。

9.3.4.1 NP 类更多的 PCP 特征

有趣的是, $\mathcal{NP}$ 类 PCP 特征的两种复杂性量度可以实现折中, 使得在两种极端情况下分别有 $\mathcal{NP} = \text{PCP}(\log, O(1))$ 和 $\mathcal{NP} = \text{PCP}(0, \text{poly})$ 。

命题 9.22 对任意 $S \in \mathcal{NP}$, 存在一个对数函数 l (即 $l \in \log$), 使得对任意满足 $0 \leq k(n) \leq l(n)$ 的整数函数 k , 有 $S \in \text{PCP}(l-k, O(2^k))$ (注意: $\text{PCP}(\log, \text{poly}) \subseteq \mathcal{NP}$)。

证明概要: 由定理 9.16, 有 $S \in \text{PCP}(l, O(1))$ 。为证明 $S \in \text{PCP}(l-k, O(2^k))$, 考虑对相应验证者的模仿, 其中尝试随机带上所有可能的 $k(n)$ -比特前缀。□

根据定理 9.16, 可以研究 NP 类 PCP 特征的大量变形。这些变形主要针对概率可验证证明系统 (对于 $\mathcal{NP}$ 中集合) 更精细的参数分析。以下简要介绍一些相关研究。^①

PCP 的长度

在任意 $\text{PCP}(\log, \log)$ 系统中, 预言的有效长度为 (关于输入长度的) 多项式。进一步地, 在证明定理 9.16 时使用的 PCP 系统中, 询问只涉及到预言的多项式长度前缀, 所以 $\mathcal{NP}$ 的这类 PCP 实际上是多项式长度的。显然可令 $\mathcal{NP}$ 的 PCP 为近乎线性长度 (输入和标准 NP-证据加起来的总长度), 而仍保持询问复杂度为常数, 其中近乎线性的意思是与线性相差一个多项式-对数因子 (细节可见参考文献[36, 67])。这意味着, 如果在证明预言中引入适量冗余, 则可以支持含常数次询问的概率验证。

PCP 中的查询次数

定理 9.16 称在具有对数随机性复杂度和 $1/2$ 的合理性误差的 (关于 NP 的) PCP 中, 常数次询问就足够了。目前已知该常数至多为 5, 而 3 次询问可以得到非常接近 $1/2$ 的合理性误差。询问次数与合理性误差间的折中可以引出更健壮的均摊 (Amortized) 询问复杂度概念, 定义为询问次数与合理性误差的 (负) 对数 (以 2 为底) 之比。对任意 $\varepsilon > 0$, $\mathcal{NP}$ 中任意集合具有对数随机性复杂度和 $1+\varepsilon$ 的均摊询问复杂度的 PCP 系统 (见参考文献[119]), 而只有 $\mathcal{P}$ 中的集合才可能有对数随机性复杂度和小于 1 的均摊询问复杂度的 PCP。

自由比特复杂度

自由比特的最初动机来自于 PCP 到最大团 (MaxClique) 的关联 (见习题 9.18 及参考文献[29]中第 8 节), 但我们认为这一概念具有独立价值。直观上, 它区分了两种情况: 一种是询问中可接受的应答由以前得到的应答确定 (即验证者比较当前应答与由以前应答确定的一个值); 另一种是询问中验证者会记录下应答以备后。后一种询问被称为是自由的 (因为任意应答都是“可接受的”)。例如, 在线性测试 (见 9.3.2.1 节) 中, 前两个询问是自由的, 而第三个不是 (即当且仅当 $f(x) + f(y) = f(x+y)$ 时, 测试才接受)。与均摊询问复杂度相似, 可以定义均摊自由比特复杂度。有趣的是, $\mathcal{NP}$ 中集合的 PCP 具有对数随机性复杂度, 然而其均摊自由比特复杂度小于任意正的常数。

适应性与非适应性验证者

称一个 PCP 验证者是非适应性的, 如果其询问只由输入和内部随机数确定 (一般性的验证者, 称为适应性的, 会根据之前收到的预言应答来确定询问)。回顾在证明 NP 的 PCP 特征时 (即定理 9.16), 使用了非适应性的验证者, 然而, 人们发现, 在一些量化结论下, 适应性

^① 除了在参考文献[90]之后出现的一些工作, 在这里没有引用任何其他参考文献。感兴趣的读者可以阅读参考文献[90]中 2.4.4 节。

的验证者比非适应性的更强大：具体地，对于发出 3 次询问及对数随机性复杂度的 PCP 验证者而言，适应性的询问可以使 NP 中任意集合的合理性误差至多为 0.51（实际上是 $0.5 + \varepsilon$ ，对任意 $\varepsilon > 0$ ），而非适应性的询问只能使 P 中任意集合的合理性误差达到 $5/8$ （或更小）。

非二进制的询问

PCP 系统的定义中只允许使用二进制询问。非二进制的询问可以由二进制询问模仿，但反之则不一定成立。^①出于这个原因，“并行重复”在 PCP 环境下是极特殊的。已知有针对于独立调用同一个 PCP 的并行重复定理，但利用该定理不能使合理性误差小于某个常数（同时保持对数的随机性复杂度）。不过，利用充分的“一致性测试”，可以为 NP 类构造具有对数随机性复杂度、常数次（非二进制）询问及合理性误差为应答长度指数的 PCP 系统（目前这只对亚对数长度应答成立。）

9.3.4.2 NP 类更强的 PCP 系统

虽然 PCP 定理主要因其在近似问题研究中的消极作用而闻名（见 9.3.3 节及 10.1.1.2 节），但对其直接和正面的应用也是吸引人的。事实上，该定理在加速普通证明的验证方面的应用尤为引人瞩目，这些证明可能涉及到一些普通断言如某个具体计算的正确性。对其进行加速需要 PCP 定理的增强版本，使得验证时间确实能减少，而不“仅仅”是降低询问复杂度。这种增强是可能的。

定理 9.23（定理 9.16 的增强版本） NP 中任意集合 S 具有对数随机性复杂度、常数询问复杂度，以及二次时间复杂度的 PCP 系统 V 。进一步地， S 中成员判定的 NP-证据可在多项式时间内转化为 V 中相应的证明预言。

9.3.2 节中已阐述了定理中“进一步”的部分（作为定理 9.16 的强化）。因此，定理 9.23 的新颖之处在于提供了二次的验证时间，而非多项式的验证时间（其中多项式可能依赖于集合 S ）。在证明定理 9.23 时，注意到，将 S 归约为 3SAT 时得到的 CNF 范式非常均匀，从而 9.3.2.2 节中描述的验证者 V 可在二次时间内实现。事实上，对 V 而言最耗时的操作是求 (C_ϕ) 中 ϕ 的低次扩展，这对应于输入范式 ϕ 中的几个点。9.3.2.2 节中，在指数时间内计算 ϕ 就足够了（因为此时计算时间为 $|\phi|$ 的多项式）。证明 ϕ 的一种变形可在多项式时间内求值，这就证明了定理 9.23（因为这意味着计算时间为 $|\phi|$ 的多项式—对数），详见习题 9.30。

近似 PCP

显然，我们不能期望一个 PCP-系统（或任意标准证明系统）具有亚线性的验证时间（因为仅仅是读取输入就需要线性时间）。然而，可以考虑验证任务的一种宽松情形（将证明当作某集合 S 的成员）。此时，只要求验证者能拒绝任意与 S “距离较远”的输入（而无需考虑证明），并且按照惯例接受任意属于 S 的输入（伴随着充分的证明）。具体地，为了实现亚线性的验证时间，为验证者 V 提供对输入比特的直接访问（被看作是预言），以及对通常（PCP）证明预言的直接访问，并要求满足以下两个条件（关于同一个常数 $\varepsilon > 0$ ）。

完备性：对任意 $x \in S$ ，存在一个串 π_x ，使得当给定对 x 和 π_x 的预言访问时，机器 V 总是接受。

① 麻烦的根源在于，（合理性条件中暗指的）敌手环境中，当几个二进制询问被打包为一个询问时，敌手并不需要参考打包过程（即可根据一起打包的询问来对同一个询问做出不一致的应答）。在 PCP 中，这个困难尤为显著，因为并不是向一个完整的信息游戏响应。相反，在交互式证明系统中，要实现并行重复十分容易，因为这种证明可被模型化为完整的信息游戏：在公开掷硬币系统中，这很显然，而在一般的交互式证明中也是成立的（见习题 9.1）。

关于近似 ε 的合理性: 对任意与 S 距离为 ε 的串 x (即对任意 $x' \in \{0,1\}^{|x|} \cap S$, x 与 x' 至少有 $\varepsilon|x|$ 个比特不同) 以及任意串 π , 当给定对于 x 和 π 的预言访问时, 机器 V 以至少 $1/2$ 的概率拒绝。

机器 V 称为近似 PCP, 其对两个预言的访问次数被计入询问复杂度中 (事实上, 9.3.2.2 节中已经使用了近似 PCP, 这个概念类似于 10.1.2 节中判定问题的宽松版本)。

我们指出, $\mathcal{NP}$ 中任意集合都具有对数随机性复杂度、常数询问复杂度以及多项式一对数时间复杂度的近似 PCP。这可利用定理 9.23 的证明思想来证明 (还可见于习题 9.30)。

9.3.4.3 具有超对数随机性复杂度的 PCP

目前为止, 只讨论了一种重要情形, 即验证者使用了对数个随机数, 从而“有效证明长度”为多项式。这里我们指出, PCP 定理 (或定理 9.23) 是按比例增加的。^①

定理 9.24 (定理 9.16 的推广) 令 $t(\cdot)$ 为整数函数, 满足 $n < t(n) < 2^{\text{poly}(n)}$, 则 $N_{\text{TIME}}(t) \subseteq \text{PCP}(O(\log t), O(1))$ 。

回顾对于 $t(n) = \text{poly}(n) \cdot 2^{r(n)}$, 有 $\text{PCP}(r, q) \subseteq N_{\text{TIME}}(t)$ 。因此, N_{TIME} 层级中蕴含了 $\text{PCP}(\cdot, O(1))$ 类的随机性复杂度从对数到多项式范围内的一个层级。

本章注释

以下对于研究背景的注释较长, 但仍没有包含在该领域发展中起重要作用的一些工作。更多的细节请见参考文献[90]中 2.6.2 节。

为了形式化地描述密码协议中使用的通用“证明”方法, Goldwasser、Micali 和 Rackoff^[109] 提出了交互式证明系统。虽然他们的工作本质上引入了一种特殊类型的交互式证明 (即零知识证明), 但参考文献[109]中还指出交互式证明可能比 NP-证明功能更强大。独立于参考文献[109]中的工作, Babai^[118] 提出了交互式证明的另一种形式化方法, 称为 Arthur-Merlin 游戏。从语法上看, Arthur-Merlin 游戏是一种受限形式的交互式证明, 后来人们证明这种受限形式与通用的交互式证明具有同等功能 (见参考文献[111])。加速结论 (即 $\mathcal{AM}(2f) \subseteq \mathcal{AM}(f)$) 来自于参考文献[23] (是参考文献[18]中工作的改进)。

关于交互式证明的功能, 第一个明显的证据由 Goldreich, Micali 和 Wigderson 给出^[100], 其中提出了图的同构问题的交互式证明系统 (构造 9.3)。更重要地, 他们论证了零知识证明的通用性及其广泛应用: 他们解决了假设单向函数存在时, 对 $\mathcal{NP}$ 中任意集合如何构造交互式零知识证明的问题 (定理 9.11)。这个结论对于密码协议的设计具有重大意义 (见参考文献[101])。附录 C 中进一步讨论了零知识证明及其在密码学中的应用。定理 9.12 (即 $\mathcal{ZK} = \mathcal{IP}$) 则来自于参考文献[32, 130]。

概率可检验的证明 (PCP) 系统与具有多个证明者的交互式证明系统有关, 这是交互式证明的一种推广, 由 Ben-Or、Goldwasser、Kilian 和 Wigderson 提出^[33]。其动机仍来自零知识的角度, 具体而言, 对 $\mathcal{NP}$ 引入了不依赖于困难性假设的多证明者零知识证明。这种新的类型, 称为 $\mathcal{MIP}$, 其在复杂性理论方面的应用也不可忽视。

① 注意: 定理 9.23 的证明概要中, 验证时间是输入长度的平方, 是 NP-证据长度的多项式-对数。

用代数方法研究交互式证明系统,可以得到一些令人惊奇的结论。这种基本技术由 Lund、Fortnow, Karloff 和 Nisan 引入^[162], 他们用代数方法证明了多项式时间层级 (实际上是 $\mathcal{P}^{\#P}$) 属于 IP 。随后, Shamir^[202]利用该技术证明了 $IP=PSPACE$, 而 Babai、Fortnow 和 Lund^[21]也利用该方法证明了 $MIP=NEXP$ (我们对定理 9.4 的证明就参考文献[202])。

上述由 Babai、Fortnow 和 Lund^[21]提出的多证明者证明系统 (以后都称为 BFL 证明系统) 曾经是关于 NP 的基础性研究进展的起点。第一个进展是发现了 BFL 证明系统可以由 $NEXP$ “缩小”到 NP 。这个重要结论是由两组研究人员独立发现的: Babai、Fortnow、Leven 和 Szegedy[20]是一组, Feige、Goldwasser、Lovász 和 Safra[73]是另一组。然而, 在两篇文章中, BFL 证明的缩小方式有所不同, 从而缩小的结果也不同。

Babai 等人^[20]考虑将输入用一种特殊的纠错码来编码, 这种对串的编码可在多项式时间内完成。他们提出了一个几乎线性时间算法将 (来自集合 $S \in NP$ 的输入的) NP -证据转化为可以验证的透明证明 (保证编码后断言的正确性), 验证的过程可在 (概率) 多项式-对数时间 (由随机存取机) 完成。Babai 等人^[20]强调了透明证明的实际作用, 具体而言, 可用于快速检查一个较长计算的副本。

相反, 在 Feige 等人^[73, 74]提出的证明系统中, 验证者一直为多项式时间, 只有两个附加的精确复杂性量度 (即随机性和询问复杂度) 被缩小至多项式-对数。这就不需要假设输入为某种特殊的纠错形式, 并得到概率可验证证明系统一种精确的 (量化) 版本 (在参考文献[80]中引入), 其中精确性通过规定随机性和询问复杂度而得到 (见定义 9.14)。因此, 尽管 BFL 证明系统^[21]可以重新解释为证明 $NEXP = PCP(\text{poly}, \text{poly})$, 但 Feige 等人的工作^[74]证明了 $NP \subseteq PCP(f, f)$, 其中 $f(n) = O(\log n \cdot \log \log n)$ (回想起来, 注意到 Babai 等人的工作蕴含了 $NP \subseteq PCP(\log, \text{polylog})$)。

自从 Feige 等人的工作^[73, 74]表明新的复杂性类可用于证明某些近似组合问题 (如最大团问题, MaxClique) 的困难性之后, 人们对新的复杂性类产生了广泛兴趣。当使用由 Feige 等证明的 PCP 到最大团间的联系时, (在 NP -完全集的 PCP 系统中) 验证者的随机性和询问复杂度与近似问题中相反结论的强度有关。这一事实促使人们尝试减小这些复杂度, 并得到 NP 类以 $PCP(., .)$ 术语表示的一个更紧凑的特征。一个明显的挑战是证明 NP 等于 $PCP(\log, \log)$ 。这个问题被 Arora 和 Safra^[16]解决。他们证明了 $NP=PCP(\log, q)$, 其中 $q(n) = o(\log n)$ 。

由此产生一个新的问题, 即能否进一步地减小小询问复杂度——特别是将其减小至一个常量——而仍保持对数随机性复杂度。这个挑战还有一个动机来自于该结论与近似问题研究的关系。这个新的问题被 Arora、Lund、Motwani、Sudan 和 Szegedy 解决^[15], 并可表示为 PCP 特征定理, 其中断言 $NP=PCP(\log, O(1))$ 。

事实上, PCP 特征定理是一系列卓越工作^[15, 16, 20, 21, 74, 162]的巅峰。这些工作中包含了大量创新思想 (如对 SAT 的各种算术化, 以及各种形式的证明组合), 并使用了各种不同技术 (如低次测试、自纠错以及随机性)。

我们对于 PCP 定理原始证明的概述 (9.3.2.1 节和 9.3.2.2 节) 基于参考文献[15, 16]。^①9.3.2.3 节中介绍的另一种证明方法由 Dinur 给出^[67]。

我们还提到在证明 PCP 定理更强的变形 (9.3.4.1 节中给出了综述) 中, 涉及到的一些思想和技术, 包括并行重复定理^[185]、长码 (Long-Code) 的使用^[29], 以及这种环境中傅里叶分

① 我们的阐述也利用了 PCP 的近似性和健壮性, 这两个概念来自于参考文献[35, 68]。

析的使用^[116, 117]。我们还强调了 PCP 近似和健壮性的概念（见参考文献[35, 68]）。

计算合理的证明系统

为了对 $\mathcal{NP}$ 提供完善的零知识论据（而非零知识证明），Brassard、Chaum 和 Crépeau 等人^[49]定义了论据系统。几年后，Kilian^[145]阐述了该系统在零知识证明之外的用途，他证明了，在某些合理的困难性假设下， $\mathcal{NP}$ 中每个集合都有计算上合理的证明，其随机性和通信复杂度均为多项式一对数。^①有趣的是，这些论据系统依赖于一个事实，即 $\mathcal{NP} \subseteq \mathcal{PCP}(f, f)$ ，其中 $f(n) = \text{poly}(\log n)$ 。另外，Micali^[165]提出了另一种类型的计算合理的证明系统（他称为 CS-证明）。

最后的说明

本章是参考文献[90]中第 2 章的修订版本。其中提供了关于主要内容的更多细节，而忽略了（或只是简单地提到）参考文献[90]中的大量次要内容。注意：我们在参考文献[90]中 2.4.4 节提到的几个研究方向在过去一段时间内引起了广泛注意，且目前也有了更好的结论。感兴趣的读者可以见参考文献[35, 36, 67]中对 PCP 长度的研究，以及参考文献[119]中对均摊询问复杂度的研究。类似地，参考文献[90]中 2.6.3 节提到的几个公开问题也已得到了解决，感兴趣的读者可以见参考文献[25, 172]中关于零知识的突破性结论。

习 题

9.1 （交互式证明系统的并行错误缩减） 对证明系统 (P, V) 的 t 次并行重复是指一个交互，其中执行基本系统的 t 个副本，使得在第 i 次动作中，相关参与方为每个副本执行第 i 次动作。当然，一个诚实的参与方（即验证者）在每个副本中的行为独立于在其他副本中的行为，但是不诚实的证明者可能会基于所有副本的执行来确定其在每个副本中的行为。尽管如此，证明此时（合理性条件中的）错误概率随着（证明系统）并行重复次数而呈指数降低。

提示：作为热身，首先考虑一种特殊情形，即公开掷硬币交互式证明系统。然后，将分析推广到任意的交互式证明系统，方法是考虑一个“有能力的验证者”，可以与公开掷硬币模型中一样，模仿原始验证者的行为（参考文献[90]中附录 C.1 介绍了一种直接证明）。

9.2 证明如果 S 可以 Karp-归约到 IP 中一个集合，则 $S \in IP$ 。证明如果 S 可以 Cook-归约到集合 S' ，使得 S' 和 $\{0, 1\}^* \setminus S'$ 属于 IP ，则 $S \in IP$ 。

9.3 完成 $\text{co}\mathcal{NP} \subseteq IP$ 的证明细节（即定理 9.4 证明的第一部分）。特别地，假设将不可满足性协议应用于含 n 个变量和 m 个子句的 CNF 范式，则双方发出的消息长度是多少？合理性错误又是多少？

9.4 构造不可满足性的交互式证明系统，使得对于输入的 n 变量 CNF 范式，双方进行 $n/O(\log n)$ 次消息交互。

提示：修改定理 9.4 的证明（第一部分），方法是分割每一轮中的 $O(\log n)$ 次求和。

9.5 （ $\#P$ 的交互式证明系统） 利用定理 9.4 证明的主要部分，构造式 (9.5) 的计数集的证明系统。

提示：利用 CNF 范式的一个稍有不同的算术化版本。具体地，子句 $x \vee \neg y \vee z$ 不是被替

① 我们指出，交互式证明不太可能有这样低的复杂度，见参考文献[106]。

换为 $(x + (1 - y) + z)$ ，而是被替换为 $(1 - ((1 - x) \cdot y \cdot (1 - z)))$ 。这种算术化可以将满足 CNF 范式的布尔赋值映射为 0-1 赋值，使得替换后的相应算术表达式取值为 1（而非任意正整数）。

9.6 证明 QBF 可以化简为一种特殊形式的（非正则的）^①QBF，其中不存在一个变量，对多于一个全称量词既出现在左边也出现在右边。

提示：考虑一个（从左向右运行的）“更新”过程，在每个全称量词后对变量进行更新。令 $\phi(x_1, x_2, \dots, x_s, y, x_{s+1}, x_{s+2}, \dots, x_{s+t})$ 为无量词的布尔公式，且令 $Q_{s+1}, Q_{s+2}, \dots, Q_{s+t}$ 为任意的量词序列。则将量化后的（子）公式

$$\forall y Q_{s+1} x_{s+1} \cdots Q_{s+t} x_{s+t} \phi(x_1, x_2, \dots, x_s, y, x_{s+1}, x_{s+2}, \dots, x_{s+t})$$

替换为（子）公式

$$\forall y \exists x'_1 \cdots \exists x'_s \left[\left(\bigwedge_{i=1}^s (x'_i = x_i) \right) \wedge Q_{s+1} x_{s+1} \cdots Q_{s+t} x_{s+t} \phi(x'_1, x'_2, \dots, x'_s, y, x_{s+1}, x_{s+2}, \dots, x_{s+t}) \right]$$

注意：在替换后的公式中，变量 $x_1, x_2, \dots, x_s$ 不出现在量词 Q_{s+1} 的右边，且替换后公式的长度增加了一个 $O(s)$ 项。这个更新变量的过程从左向右地应用于整个全称量词序列（除了内部的一个，更新过程对其不起作用）。^②

9.7 证明如果 $[0, M]$ 中两个整数是不同的，则它们模 $[3, L]$ 中大部分素数都不同，其中 $L = \text{poly}(\log M)$ 。对于区间 $[L, 2L]$ ，证明同样的结论成立。

提示：令 $a \neq b \in [0, M]$ ，并假设 $P_1, P_2, \dots, P_t$ 为所有满足 $a \equiv b \pmod{P_i}$ 的素数。利用中国剩余定理，证明 $Q = \prod_{i=1}^t P_i \leq M$ （因为结合 $a \equiv b \pmod{Q}$ 与 $a, b \in [0, M]$ 的假设，可以得到 $a = b$ ），从而有 $t < \log_2 M$ 。利用素数分布的密度下限，断言得证。

9.8（具有双边错误的交互式证明（由参考文献[78]可知））令 $IP'(f)$ 表示具有双边错误的交互式证明系统集合，其中对于公共输入 x ，要交互的消息总数为 $f(|x|)$ 。具体地，假设一个适当的证明者使每个“是”实例以至少 $2/3$ （而不是 1 ）的概率被接受，而不诚实的证明者不可能使“否”实例被接受的概率大于 $1/3$ （而非 $1/2$ ）。类似地，令 AM' 表示 IP' 的公开掷硬币版本。

（1）利用定理 F.2 的方法来证明 $IP'(f) \subseteq AM'(f+3)$ ，其中证明了 $IP(f) \subseteq AM(f+3)$ ，将这个结论扩展到双边错误情况。

（2）对定理 6.9 的证明思想进行推广，证明 $AM'(f) \subseteq AM(f+1)$ ，定理 6.9 的证明中实际上证明了 $BPP \subseteq AM(1)$ （其中 $BPP = AM'(0)$ ）。

利用轮加速定理（即定理 F.3）得到结论，对任意函数 $f: \mathbb{N} \rightarrow \mathbb{N} \setminus \{1\}$ ，有 $IP'(f) = AM(f) = IP(f)$ 。

提示（第二部分）：对于一种给定的双边错误公开掷硬币的交互式证明，固定优化的证明者策略，考虑验证者的一组随机数集合，使验证者接受任意一个给定的“是”实例，并利用从 BPP 到 $MA = AM(1)$ 的变换思想。进一步的细节见参考文献[82]。

① 见附录 G.2。

② 例如， $\exists z_1 \forall z_2 \exists z_3 \forall z_4 \exists z_5 \forall z_6 \phi(z_1, z_2, z_3, z_4, z_5, z_6)$ ，先被替换为 $\exists z_1 \forall z_2 \exists z'_1 \left[(z'_1 = z_1) \wedge \exists z_3 \forall z_4 \exists z_5 \forall z_6 \phi(z'_1, z_2, z_3, z_4, z_5, z_6) \right]$ 然后（写作 $\exists z_1 \forall z'_2 \exists z'_1 \left[(z'_1 = z_1) \wedge \exists z'_3 \forall z'_4 \exists z'_5 \forall z'_6 \phi(z'_1, z'_2, z'_3, z'_4, z'_5, z'_6) \right]$ ）被替换为 $\exists z_1 \forall z'_2 \exists z'_1 \left[(z'_1 = z_1) \wedge \exists z'_3 \forall z'_4 \exists z'_5 \exists z'_6 \left[\left(\bigwedge_{i=1}^3 (z'_i = z_i) \right) \wedge \exists z'_6 \phi(z'_1, z'_2, z'_3, z'_4, z'_5, z'_6) \right] \right]$ 因此，在得到的等式中，不存在这样的变量，对多于一个的全称量词，既出现在左边又出现在右边。

9.9 续习题 9.8, 证明对任意函数 $f: \mathbf{N} \rightarrow \mathbf{N}$ (包括 $f \equiv 1$), 有 $IP'(f) = IP(f)$ 。

提示: 主要证明 $IP'(1) = IP(1)$, 这与习题 6.12 的第二部分相同。注意到, 习题 6.12 中定义的相关类与 $IP(1)$ 和 $IP'(1)$ 是一致的, 即 $MA = IP(1)$ 且 $MA^{(2)} = IP'(1)$ 。

9.10 证明任意的 $PSPACE$ -完全集 S 具有一个交互式证明系统, 其中的证明者可以利用能访问 S 的概率多项式时间预言机实现。

提示: 利用定理 9.4 和命题 9.5。

9.11 (检查者(由参考文献[39]可知))一个概率多项式时间预言机 C 被称为判定问题 Π 的检查者, 如果以下两个条件满足:

(1) 对任意 x , 有 $\Pr[C^\Pi(x) = 1] = 1$, 其中(照例)用 $C^f(x)$ 表示给定 f 的预言访问时, C 对于输入 x 的输出。

(2) 对任意满足 $f(x) \neq \Pi(x)$ 的 $f: \{0,1\}^* \rightarrow \{0,1\}$ 及 x , 有 $\Pr[C^f(x) = 1] \leq 1/2$ 。

注意: 当 $f(x) = \Pi(x)$ 但 $f \neq \Pi$ 时, 没有任何要求。证明如果 $S_1 = \{x: \Pi(x) = 1\}$ 和 $S_0 = \{x: \Pi(x) = 0\}$ 都具备交互式证明系统, 其中证明者用能访问 Π 的概率多项式时间预言机实现, 则 Π 有一个检查者。利用习题 9.10 得到结论, 任意 $PSPACE$ -完全问题都有一个检查者。

提示: 对于输入 x 和到 f 的预言访问, 检查者首先得到 $\sigma = f(x)$ 。在检查断言 $\Pi(x) = \sigma$ 时, 可以结合 S_σ 的验证者与描述证明者的概率多项式时间预言机, 并参考到 f 的预言询问。

9.12 (可检查问题的弱优化判定者(根据参考文献[133])) 证明如果判定问题 Π 具有检查者(定义见习题 9.11), 则存在概率算法 A , 满足以下两个条件:

(1) A 解判定问题 Π (即对任意 s , 有 $\Pr[A(x) = \Pi(x)] \geq 2/3$)。

(2) 对任意解判定问题 Π 的概率算法 A' , 存在一个多项式 p , 使得对任意 x , 有 $t_A(x) = p(|x|) \cdot \max_{|x'| \leq p(|x|)} \{t_{A'}(x')\}$, 其中 $t_A(z)$ (或 $t_{A'}(z)$) 表示 A (或 A') 对于输入 z 的计算步数。

注意: 与定理 2.33 相比, 这里对于最优性的断言较弱, 但是, 另一方面, 它可以应用于判定问题(而不是公正搜索问题)。

提示: 利用定理 2.33 的证明思想, 注意到, 可以利用检查者来验证各种候选算法提供答案的正确性。也就是说, A 调用检查者的副本, 并在不同的副本中使用不同候选算法作为预言。

9.13 (零知识证明中合理性误差的作用) 证明如果 S 具有完善合理(即合理性误差为 0)

的交互式零知识证明系统, 则 $S \in BPP$ 。

提示: 令 M 为任意一个算法, 可以模仿(诚实)验证者的观察。考虑如下算法, 对输入 x , 算法接受 x 当且仅当 $M(x)$ 表示验证者在一次接受交互(使验证者接受公共输入 x 的交互)中的有效观察。利用模拟条件分析 $x \in S$ 的情形, 并利用完善合理假设分析 $x \notin S$ 的情形。

9.14 (零知识证明中交互的作用) 证明如果 S 具有一个单向通信的交互式零知识证明系统, 则 $S \in BPP$ 。

提示: 令 M 为模仿(诚实)验证者观察的算法, 并令 $M'(x)$ 表示观察中包含证明者消息的部分。考虑一个算法, 对于输入 x , 得到 $m \leftarrow M'(x)$, 并模仿验证者对于输入 x 和消息 M 的判定。注意: 该算法忽略了 $M(x)$ 中代表验证者内部随机数的部分, 并在判定 (x, m) 时使用

新鲜的验证者随机数。

9.15 (PCP 预言的有效长度) 假设 V 为具有询问复杂度 q 和随机性复杂度 r 的 PCP 验证者。证明 V 对于任意固定的输入 x , 在证明预言中检查的位置数量 (考虑到 V 的所有可能的内部随机数以及 V 接收到的所有可能应答时) 上限为 $2^{q(|x|)+r(|x|)}$ 。证明如果 V 是非适应性的, 则该上限可以提高至 $2^{r(|x|)} \cdot q(|x|)$ 。

提示: 在非适应性情况下, 所有的 q 次询问都由 V 的内部随机数确定。

9.16 (PCP 的高效随机性) 假设集合 S 具有询问复杂度为 q 的 PCP, 其中使用长为 l 的证明预言。证明对任意常数 $\varepsilon > 0$, 集合 S 具有 “非一致的” PCP, 其询问复杂度为 q , 合理性误差为 $0.5 + \varepsilon$, 且随机性复杂度为 r , 满足 $r(n) = \log_2(l(n) + n) + O(1)$ 。这里 “非一致的 PCP” 是指验证者为概率多项式时间预言机, 给定了到非一致的 $\text{poly}(l(n) + n)$ -比特建议的直接访问。

提示: 考虑假设中的 PCP 验证者 V , 将其随机性复杂度记作 r_V 。我们构造一个非一致验证者 V' , 对于长为 n 的输入, 得到含 $O((l(n) + n)/\varepsilon^2)$ 个元素的集合 $R_n \subseteq \{0, 1\}^{r(n)}$ 作为建议, 并对 R_n 中均匀选择的元素模仿 V 。证明对于这样的随机 R_n , 验证者 V' 满足习题中断言。

(附加提示: 给定任意输入 $x \notin S$, 以及任意一个预言 $\pi \in \{0, 1\}^{l(|x|)}$, 求随机集合 R_n (具有上述元素个数) 为 “坏” 集合的概率上限, 其中称 R_n 为 “坏” 的, 如果 V 从 R_n 中选择随机数, 并利用预言 π 时, 以概率 $0.5 + \varepsilon$ 接受 x 。)

9.17 (具有某种 PCP 的集合的复杂性) 假设集合 S 具有询问复杂度 q 和随机性复杂度 r 的 PCP。证明 S 可以用一个非确定机器来判定^①, 对于长为 n 的输入, 最多运行 $2^{r(n)} \cdot q(n)$ 个真正非确定的步骤 (即需要在不同选项中进行选择), 并在运行了 $2^{r(n)} \cdot \text{poly}(n)$ 步后停机。得到结论, $S \in N_{\text{TIME}}(2^r \cdot \text{poly}) \cap D_{\text{TIME}}(2^{2^r q + r} \cdot \text{poly})$ 。

提示: 对每个输入 $x \in S$ 及验证者随机带上每个可能的值 $\omega \in \{0, 1\}^{r(|x|)}$, 考虑一个由 $q(|x|)$ 比特构成的序列, 该序列表示使验证者接受的一系列预言应答。事实上, 对于固定的 x 和 $\omega \in \{0, 1\}^{r(|x|)}$, $q(|x|)$ 个预言应答确定了验证者的相应计算 (包括其发出的询问)。

9.18 (FGLSS-归约^[74]) 对任意 $S \in \text{PCP}(r, q)$, 考虑以下由 S 的实例到独立集 (Independent Set) 问题实例的映射。实例 x 被映射为图 $G_x = (V_x, E_x)$, 其中 $V_x \subseteq \{0, 1\}^{r(|x|)+q(|x|)}$ 由一些对 (ω, α) 组成, 使得 PCP 验证者在使用随机数 $\omega \in \{0, 1\}^{r(|x|)}$, 并收到应答 $\alpha = \alpha_1 \alpha_2 \cdots \alpha_{q(|x|)}$ (针对由 x, r 和以前应答确定的预言询问) 时接受输入 x 。注意: V_x 中只包含验证者接受的 “观察”。集合 E_x 中的边连接着代表验证者不一致观察的结点, 也就是说, 如果存在 i 和 i' , 使得 $\alpha_i \neq \alpha'_{i'}$ 且 $q_i^x(v) = q_{i'}^{x'}(v')$, 则结点 $v = (\omega, \alpha_1 \alpha_2 \cdots \alpha_{q(|x|)})$ 与 $v' = (\omega', \alpha'_1 \alpha'_2 \cdots \alpha'_{q(|x|)})$ 之间有边相连, 其中 $q_i^x(v)$ (或 $q_{i'}^{x'}(v')$) 表示验证者对于输入 x 的第 i 个 (或第 i' 个) 询问, 其中使用的随机数为 ω (或 ω'), 且收到的应答为 $\alpha_1 \alpha_2 \cdots \alpha_{i-1}$ (或 $\alpha'_1 \alpha'_2 \cdots \alpha'_{i-1}$)。特别地, 对任意 $\omega \in \{0, 1\}^{r(|x|)}$ 及

^① 非确定机的定义见 4.2.1.3 节。

$\alpha \neq \alpha'$, 如果 $(\omega, \alpha), (\omega, \alpha') \in V_x$, 则 $\{(\omega, \alpha), (\omega, \alpha')\} \in E_x$ 。

(1) 证明映射 $x \mapsto G_x$ 可在 $2^{r(|x|)+q(|x|)} \cdot |x|$ 的多项式时间内计算。

(注意: G_x 的结点数上限为 $2^{r(|x|)+f(|x|)}$, 其中 $f \leq q$ 是 PCP 验证者的自由比特复杂度。)

(2) 证明对任意 x , G_x 中的最大独立集规模不超过 $2^{r(|x|)}$ 。

(3) 证明如果 $x \in S$, 则 G_x 具有规模为 $2^{r(|x|)}$ 的独立集。

(4) 证明如果 $x \notin S$, 则 G_x 的最大独立集规模不超过 $2^{r(|x|)-1}$ 。

一般将 PCP 验证者记作 V , 证明 G_x 的最大独立集规模恰为 $2^{r(|x|)} \cdot \max_{\pi} \{\Pr[V^{\pi}(x)=1]\}$ (注

意: 与命题 2.25 证明的相似性。)

证明 PCP 定理蕴含了在常数因子范围内近似计算图中最大独立集 (或团) 的规模是 NP-困难的。

提示: 注意: G_x 中的独立集对应于随机数集合 R 和部分预言 π' , 满足 V 使用 R 中随机数并访问任何与 π' 一致的预言时, 接受 x 。FGLSS-归约在 S (具有标准 PCP) 的“是”实例和“否”实例之间形成一个因子为 2 的缝隙。重复原始 PCP 常数次, 可得到更大的因子。如果考虑完全图, 则可以得到关于团的结论。

9.19 利用习题 9.18 的思想, 证明对任意 $t(n) = o(\log n)$, 如果 $\mathcal{NP} \subseteq \text{PCP}(t, t)$, 则 $\mathcal{P} = \mathcal{NP}$ 。

提示: 只利用一个事实, 即 FGLSS-归约将 $S \in \text{PCP}(t, t)$ 的实例映射为团的实例 (并忽略我们实际上得到了到“缝隙团”问题的一个更强归约)。关键在于, 当应用于 $\text{PCP}(t, t)$ 中问题的一个 n 比特实例时, FGLSS-归约运行于多项式时间, 并生成规模为 $2^{2t(n)} \ll n$ 的实例。从而, $\mathcal{NP} \subseteq \text{PCP}(t, t)$ 的假设蕴含了 FGLSS-归约将团的实例映射为同一问题的更短实例。因此, 重复应用 FGLSS-归约, 可将团的实例映射为具有常数规模的实例。这就得到了团到有限集的归约, 从而 $\mathcal{NP} = \mathcal{P}$ 得证 (由团的 $\mathcal{NP}$ -完全性)。

9.20 (对 BLR 线性测试一个简单的部分分析) 对于可交换群 G 和 H , 考虑从 G 到 H 的函数。对于这样的 (普通) 函数 f , 考虑一种线性 (或同态性) 测试, 均匀选择 $r, s \in G$, 并检查是否有 $f(r) + f(s) = f(r+s)$ 。令 $\delta(f)$ 表示 f 与 (G 到 H 的) 同态集的距离, 即 $\delta(f)$ 是所有同态映射 $h: G \rightarrow H$ 中概率 $\Pr_{x \in G}[f(x) \neq h(x)]$ 的最小值。利用以下提示, 证明测试中拒绝 f 的概率, 记作 $\varepsilon(f)$, 至少为 $3\delta(f) - 6\delta(f)^2$ 。

(1) 假设 h 是与 f 最近的同态 (即 $\delta(f) = \Pr_{x \in G}[f(x) \neq h(x)]$)。证明 $\varepsilon(f) = \Pr_{x, y \in G}[f(x) + f(y) \neq f(x+y)]$ 的下界为

$$3 \cdot \Pr_{x, y}[f(x) \neq h(x) \wedge f(y) = h(y) \wedge f(x+y) = h(x+y)]$$

(提示: 当 $f(x) + f(y) \neq f(x+y)$ 时, 考虑 4 种可能情况中的 3 种 (关于 $f(x) \stackrel{?}{=} h(x)$, $f(y) \stackrel{?}{=} h(y)$, 以及 $f(x+y) \stackrel{?}{=} h(x+y)$), 这 3 种情况对应着 h 与 f 恰好在 3 个相关点之一不一致。

(2) 证明 $\Pr_{x, y}[f(x) \neq h(x) \wedge f(y) = h(y) \wedge f(x+y) = h(x+y)] \geq \delta(f) - 2\delta(f)^2$ 。

(提示: 对该概率求下界为

$$\Pr_{x, y}[f(x) \neq h(x)] - \left(\Pr_{x, y}[f(x) \neq h(x) \wedge f(y) \neq h(y)] + \Pr_{x, y}[f(x) \neq h(x) \wedge f(x+y) \neq h(x+y)] \right)$$

注意: 仅当 $\delta(f) \leq 1/4$ 时, 下界 $\varepsilon(f) \geq 3\delta(f) - 6\delta(f)^2$ 随 $\delta(f)$ 而增加。进一步地, 当 $\delta(f) \leq 1/2$ 时, 该下界毫无意义。因此, 当 $\delta(f)$ 趋近于 (或大于) $1/2$ 时, 需要另一个下界, 见习题 9.21。

9.21 (对 BLR 线性测试的更全面分析 (见参考文献[140])) 续习题 9.20, 按顺序使用以下提示, 证明 $\varepsilon(f) \geq \min(1/6, \delta(f)/2)$ 。具体地, 主要考虑 $\varepsilon(f) < 1/6$ 的情形, 证明 f 与某个同态是 $2\varepsilon(f)$ -接近的 (从而 $\varepsilon(f) \geq \delta(f)/2$)。

(1) 定义 y 关于 f 在 x 取值的表决为 $\phi_y(x) \stackrel{\text{def}}{=} f(x+y) - f(y)$, 并定义 $\phi(x)$ 为相应的胜出票数 (即 $\phi(x) \stackrel{\text{def}}{=} \arg \max_{v \in H} \{ | \{ y \in G : \phi_y(x) = v \} | \}$)。

证明对任意 $x \in G$, 有 $\Pr_y [\phi_y(x) = \phi(x)] \geq 1 - 2\varepsilon(f)$ 。

附加提示: 给定 x , 称一个对 (y_1, y_2) 为良好的, 如果 $f(y_1) + f(y_2 - y_1) = f(y_2)$ 且 $f(x + y_1) + f(y_2 - y_1) = f(x + y_2)$ 。证明对任意 x , 一个随机对 (y_1, y_2) 为良好对的概率至少是 $1 - 2\varepsilon(f)$ 。另一方面, 一个良好对 (y_1, y_2) 满足 $\phi_{y_1}(x) = \phi_{y_2}(x)$ 。证明如果一个图中的边对应着良好对, 则该图必有一个规模至少为 $(1 - 2\varepsilon(f)) \cdot |G|$ 的连通分量。注意: 在该连通分量中, $\phi_y(x)$ 对所有的 y 都相同, 且其中包含了 G 中大部分的 y 。

(2) 证明 ϕ 是一个同态, 即证明对任意 $x, y \in G$ 。有 $\phi(x) + \phi(y) = \phi(x + y)$ 。

附加提示: 利用一个假设的表达式 $p_{x,y} \stackrel{\text{def}}{=} \Pr_{r \in G} [\phi(x) + \phi(y) = \phi(x + y)]$ 证明 $\phi(x) + \phi(y) = \phi(x + y)$ 成立, 并证明 $p_{x,y} < 1$ (从而 $\phi(x) + \phi(y) \neq \phi(x + y)$ 不成立)。证明 $p_{x,y} < 1$ 时, 可以先证

$$p_{x,y} \leq \Pr_r \left[\begin{array}{l} \phi(x) \neq f(x+r) - f(r) \\ \vee \phi(y) \neq f(r) - f(r-y) \\ \vee \phi(x+y) \neq f(x+r) - f(r-y) \end{array} \right] \quad (9.10)$$

并利用第一项 (及某些变量替换) 令式 (9.10) 中每个事件发生的概率上界为 $2\varepsilon(f) < 1/3$ 。

(3) 证明 f 与 ϕ 是 $2\varepsilon(f)$ -接近的。

附加提示: 令 $B = \{x \in G : \Pr_{y \in G} [f(x) \neq \phi_y(x)] \geq 1/2\}$, 证明 $\varepsilon(f) \geq (1/2) \cdot (|B|/|G|)$ 。注意: 如果 $x \in G \setminus B$, 则 $f(x) = \phi(x)$ 。

我们要指出, 已知 $\varepsilon(f)$ 有一个更好的上界, 是 $\delta(f)$ 的函数。

9.22 (矩阵相等的测试) 令 M 为 $\text{GF}(p)$ 上非零的 $m \times n$ 矩阵, 证明 $\Pr_{r,s} [r^T M s \neq 0] \geq (1 - p^{-1})^2$, 其中 r (或 s) 为随机的 m 维 (或 n 维) 矢量。

提示: 证明如果 $v \neq 0^n$, 则 $\Pr_s [v^T s = 0] = p^{-1}$, 而当 M 的秩为 k 时, $\Pr_r [r^T M = 0^n] = p^{-k}$ 。

9.23 (低次测试 (由参考文献[195]可知)) 对于素域 F , 整数 m 及 $d < |F| - 1$, 考虑集合 $P_{m,d}$, 其中包含 F 上所有次数不超过 d 的 m 变量多项式。低次测试的定义如下: 当给定到任意函数 $f: F^m \rightarrow F$ 的预言访问时, 均匀选择 $\bar{x}, \bar{y} \in F^m$, 在点 $(\bar{x} + i \cdot \bar{y})_{i=0,1,\dots,d+1}$ 向 f 发出询问,

当且仅当 $\sum_{i=0}^{d+1} \alpha_i f(\bar{x} + i \cdot \bar{y}) = 0$ 时接受测试, 其中 $\alpha_i = (-1)^{i+1} \cdot \binom{d+1}{i}$ 。众所周知 (见参考文献 [195]), $f \in P_{m,d}$ 当且仅当对任意 $\bar{x}, \bar{y} \in F^m$, 有 $\sum_{i=0}^{d+1} \alpha_i f(\bar{x} + i \cdot \bar{y}) = 0$ 。

(1) 根据习题 9.20, 证明测试拒绝 f 的概率至少为 $(d+2) \cdot \delta(f) - (d+2)(d+1) \cdot \delta(f)^2$, 其中 $\delta(f) = \min_{g \in P_{m,d}} \left\{ \Pr_{\bar{x} \in F^m} [f(\bar{x}) \neq g(\bar{x})] \right\}$ 。

(2) 概括习题 9.21, 证明 $\varepsilon(f) \geq \min((d+2)^{-2}, \delta(f))/2$, 其中 $\varepsilon(f)$ 表示测试拒绝 f 的概率。即证明如果 $\varepsilon(f) < (d+2)^{-2}/2$, 则 f 与 $P_{m,d}$ 中某个函数是 $2\varepsilon(f)$ -接近的。

提示: 定义 $\phi_y(x) = \sum_{i=1}^{d+1} \alpha_i f(\bar{x} + i \cdot \bar{y})$, 并注意到 $\varepsilon(f) = \Pr_{\bar{x}, \bar{y} \in F^m} [f(\bar{x}) \neq \phi_{\bar{y}}(\bar{x})]$ 。证明第一部分时, 可对这样一个事件的概率求下界: 对随机的 $\bar{x}, \bar{y} \in F^m$, 存在唯一的 $i \in \{0, 1, \dots, d+1\}$, 使得 $f(\bar{x} + i \cdot \bar{y}) \neq g(\bar{x} + i \cdot \bar{y})$, 其中 $g \in P_{m,d}$ 是与 f 最接近的低次多项式。证明第二部分时, 定义 $\phi(\bar{x}) = \arg \max_{v \in F} \left\{ \left| \left\{ \bar{y} \in F^m : \phi_{\bar{y}}(\bar{x}) = v \right\} \right| \right\}$, 其余部分类似于习题 9.21 证明中的 3 个步骤。例如, 与第一步相似, 证明对任意 $\bar{x} \in F^m$, 有 $\Pr_{\bar{y} \in F^m} [\phi(\bar{x}) = \phi_{\bar{y}}(\bar{x})] \geq 1 - 2(d+1) \cdot \varepsilon(f)$ 。

(附加提示: 证明 $\Pr_{\bar{y}_1, \bar{y}_2 \in F^m} [\phi_{\bar{y}_1}(\bar{x}) = \phi_{\bar{y}_2}(\bar{x})] \geq 1 - 2(d+1) \cdot \varepsilon(f)$ 。)^①

9.24 (3SAT 和含两个变量的 CSP) 证明 3SAT 可归约为 $\text{gapCSP}_{\frac{1}{\tau}}^{\{1,2,\dots,7\}}$, 其中 $\tau(m) = 1/m$, gapCSP 的定义见定义 9.18。进一步地, 证明得到的 gapCSP 实例的规模是输入公式长度的线性函数。

提示: 给定 3SAT 的实例 ψ , 考虑这样的图, 其中顶点对应于 ψ 中子句, 边对应于共享一个变量的子句对, 约束条件表示关于使子句满足的部分赋值的一致性条件。见习题 9.18 中的相关构造。

9.25 (含两个布尔变量的 CSP) 与习题 9.24 相反, 证明对任意正函数 $\tau: \mathbb{N} \rightarrow (0, 1]$, $\text{gapCSP}_{\frac{1}{\tau}}^{\{0,1\}}$ 可在多项式时间内求解。

提示: 将 $\text{gapCSP}_{\frac{1}{\tau}}^{\{0,1\}}$ 归约为 2SAT。

9.26 证明 对任意有限的 Σ 及常数 $c > 0$, 问题 $\text{gapCSP}_{\frac{1}{c}}^{\Sigma}$ 属于 $\text{PCP}(\log, O(1))$ 。

提示: 考虑一个预言, 对于 CSP 实例 (G, Φ) 的某个可满足赋值, 给出对该赋值一种平凡的编码。也就是说, 对于一个可满足赋值 $\alpha: V \rightarrow \Sigma$, 预言对询问 (v, i) 的应答是 $\alpha(v)$ 二元表示中的第 i 比特。考虑一个验证者, 均匀选择 G 中一条边 (u, v) , 并对得到的预言应答 $\alpha(u)$ 和 $\alpha(v)$, 验证条件 $\phi_{(u,v)}$ 是否满足。验证者需要发出 $\log_2 |\Sigma|$ 次试问, 并以至少 c 的概率拒绝每个“否”实例。

9.27 对任意常数 Σ 及 $d \geq 14$, 证明 $\text{gapCSP}_{\frac{1}{d}}^{\Sigma}$ 可归约到自身, 使得归约的目标实例是

① 以下概率表达式涉及到均匀分布的 $\bar{y}_1, \bar{y}_2 \in F^m$ 。注意: $\phi_{\bar{y}_1}(\bar{x}) = \sum_{i=1}^{d+1} \alpha_i f(\bar{x} + i \cdot \bar{y}_1)$, 它与 $\sum_{i=1}^{d+1} \alpha_i \phi_{\bar{y}_2}(\bar{x} + i \cdot \bar{y}_1)$ 相等的概率至少是 $1 - (d+1) \cdot \varepsilon(f)$ 。后者又等于 $\sum_{i_1=1}^{d+1} \sum_{i_2=1}^{d+1} \alpha_{i_1} \alpha_{i_2} f(\bar{x} + i_1 \cdot \bar{y}_1 + i_2 \cdot \bar{y}_2) = \sum_{i_2=1}^{d+1} \alpha_{i_2} \phi_{\bar{y}_1}(\bar{x} + i_2 \cdot \bar{y}_2)$ 它与 $\sum_{i_2=1}^{d+1} \alpha_{i_2} f(\bar{x} + i_2 \cdot \bar{y}_2) = \phi_{\bar{y}_2}(\bar{x})$ 相等的概率至少是 $1 - (d+1) \cdot \varepsilon(f)$ 。

一个 d -正则扩张图, 且不满足的条件比例控制在一个常数因子范围内。也就是说, 将实例 (G, Φ) 归约为 (G_1, Φ_1) , 使得 G_1 为一个 d -正则扩张图, 且 $\text{vlt}(G_1, \Phi_1) = \Theta(\text{vlt}(G, \Phi))$ 。进一步地, 确保 $|G_1| = O(|G|)$, 而 G_1 中每个结点至少有 $d/2$ 个自环。

提示: 首先, 将度数 $d' > 3$ 的每个结点替换为一个规模为 d' 的 3-正则扩张图, 并将原始的 d' 条边与该扩张图中不同的结点相连, 从而得到一个最大度数为 4 的图。保持与原始边相关的约束条件, 并将等价条件 (即 $\phi(\sigma, \tau) = 1$ 当且仅当 $\sigma = \tau$) 与每条新的边 (位于任意一个增加的扩张中) 相关联。然后, 向得到的含 N_1 个结点的图中增加规模为 N_1 的 3-正则扩张图中的边 (并将这些边与一个显然满足的条件相关联, 即对所有 $\sigma, \tau \in \Sigma$, 有 $\phi(\sigma, \tau) = 1$)。最后, 为每个结点至少增加一条自环 (再次利用显然满足的条件), 从而得到一个 d -正则图。证明这一系列的修改只可能降低不满足条件的比例, 且只降低一个常数因子。后一个断言是根据与第一步中使用的小扩张图相关联的等价条件得到的。

9.28 (自由比特复杂度为零) 注意: 只有 coRP 中集合才可能具有询问复杂度为零的 PCP。进一步地, 习题 9.17 蕴含了只有 $\mathcal{P}$ 中的集合才有对数随机性及零询问复杂度的 PCP 系统。

(1) 证明只有 $\mathcal{P}$ 中集合才有对数随机性及自由比特复杂度为零的 PCP 系统。

提示: 考虑将 FGLSS-归约应用于一个自由比特复杂度为零的 PCP 的集合。

(2) 相反, 证明非同构图问题具有自由比特复杂度为零的 PCP 系统 (以及线性的随机性复杂度)。

9.29 (自由比特复杂度为 1) 续习题 9.28, 证明只有 $\mathcal{P}$ 中集合才有对数随机性及自由比特复杂度为 1 的 PCP 系统。

提示: 考虑将 FGLSS-归约应用于具有自由比特复杂度为 1 以及随机性复杂度为 r 的 PCP 的集合。注意: 得到的图是否具有规模为 2^r 的独立集, 这一问题可以表示为具有规模为 $\text{poly}(2^r)$ 的 2CNF 范式, 见习题 2.22。

9.30 (证明定理 9.23) 利用以下提示证明定理 9.23。令 $S \in \mathcal{NP}$, 考虑由 S 到 3SAT 的标准归约得到的 3CNF 范式 (即由定理 2.21 和定理 2.22 证明中得到的)。将得到的 3CNF 范式分成一对公式 (ψ_x, ϕ) , 使得 ψ_x 表示输入 x 的“硬嵌入”, 而 ϕ 表示计算本身。参考 9.3.2.2 节中提出的 3CNF 到低次扩展的映射, 证明 Φ 的对应于 ϕ 的低次扩展可在多项式时间内计算 (即是 Φ 输入长度的多项式, 而 Φ 的输入长度为 $O(\log|\phi|)$)。得到结论, 对应于 $\psi_x \wedge \phi$ 的低次扩展可在时间 $|x|^2$ 内计算。另外, 注意到, 只需证明满足 Φ 的赋值预言机 A (9.3.2.2 节中讨论过) 与 x 是一致的 (从而是一个低次多项式)。

提示: 注意: 在定理 2.21 证明中构造的电路是强一致的。特别地, 该电路中连线与门之间的关系可以用具有常数深度、无界扇入和多项式规模 (即电路规模是连线和门指标长度的多项式) 的电路来表示。

第 10 章

对复杂性要求的弱化

哲学家们只是以不同的方式解释世界，而问题在于如何改变世界。

卡尔·马克思，关于费尔巴哈的提纲

鉴于大量的计算问题显然不可解，人们很自然地想到能否弱化这些问题的条件，使得问题仍有意义，且还存在可行的解法。我们强调两个方面：一方面，这种弱化对于相关应用来说应该足够好；另一方面，它应该与问题的原始形式明显不同，从而避免仍不可解。要注意，一种弱化对于某种应用来说是否充分取决于具体应用，因此与没有弱化的计算问题相比，本章中许多内容不是很健壮（或通用）。

内容提要：考虑两种类型的弱化。第一种类型针对着计算问题自身，即对于每个问题实例，扩展其允许的解集。在搜索问题中，这意味着求一些“足够接近”最优解（关于某个赋值函数）的解。当然，这里的“足够接近”属于弱化后问题的定义。在判定问题中，这意味着对某些实例而言，两种回答都被认为是正确的。具体地，考虑这样一类承诺问题，其中“否”实例在某种充分的意义下（包含于弱化后问题定义中）远离“是”实例。

第二种类型的弱化不要求为每个有效实例提供充分的解答。相反，是根据某种预先确定的输入分布（或一类这种分布）来分析求解者的行为，而“不好”的行为以可忽略的概率出现，其概率服从该分布。也就是说，用平均情况（或典型情况）下的分析来取代最坏情况分析。当然，这种方法的关键是要限制分布类型，使得一方面要允许各种类型的自然分布，另一方面，防止相应的平均情况困难性退化为标准的最坏情况困难性。

内容组织：10.1 节讨论第一种类型的弱化，包括搜索（或优化）问题的近似，以及近似判定问题（亦称性能测试），分别见 10.1 节及 10.1.2 节。10.2 节讨论第二种类型的弱化，也叫作平均/典型情况-复杂度。两种类型的研究方法完全不同。10.1 节总体上介绍了各种研究领域，重点介绍概念并通过一些结论（不加证明地）来阐述这些定义。10.2 节则主要介绍一个理论（即平均/典型情况-复杂度），重点是基本结论，并详述证明过程。

10.1 近 似

近似是一个很自然的概念，也用于其他学科中。在定量地描述问题时近似最为常见（如“1m 约等于 40in”），但在定性地描述问题时也比较常用（如“一个历史事件的近似正确记录”）。

在计算环境中,近似要求对搜索或判定的计算任务进行修改(事实上,我们已经遇到了对计数问题的修改,见6.2.2节)。

关于近似,有两个主要问题:什么是“好的”近似;求近似是否比求精确解更容易。第一个问题与特定的计算任务以及在更广泛环境(即更高级应用)中近似所起的作用紧密相关:一个好的近似足以满足预期应用。在重要性上,近似问题比相应优化问题更具主观性。这一事实影响着人们针对自然的近似问题(如已证明在计算上等价的近似问题类)提出综合性的复杂理论。

转向第二个问题,在许多情况下,近似问题明显比相应的原始(“精确”)问题更简单。另一方面,在大量其他情况下,近似问题在计算上等价于原始问题。我们通过几个具体结论来说明两种情况,但并不提供一种通用的系统分类(因为这样的分类尚且未知)^①。

以下将区分“搜索类型”的近似问题和显然具备“判定特性”的近似问题。第一种情况涉及对(搜索问题)可能解赋值的函数;第二种情况则涉及到(判定问题)实例间的距离。^②注意:在某些情况下,同一个计算问题可能会用两种方式处理,但是大多数近似问题只倾向于其中的一种。两种框架都采用的一个基本做法是扩大解集。对于搜索问题,将最优解集扩展为也包含几乎最优的解。在判定问题中,将解集扩展为允许对某些实例有任意的回答(解),从而可视为不接受这些实例的承诺问题。此时,重点讨论这样一类承诺问题,其中“是”实例和“否”实例距离较远(而违背承诺的实例与“是”实例距离较近)。

教学注释:本节的大部分结论都针对具体计算问题,而且没有给出证明(除了一个例外)。鉴于这些证明的复杂性以及复杂性理论中只需简单地阐述结论,建议在教学中也不加证明。

10.1.1 搜索或优化

2.2.2节提到,许多搜索问题涉及到一组可用的解集(每个实例有一个解集),使得可以通过某个“赋值”(或“估价”)函数来为不同解分配不同的“值”(或“代价”)。相应的近似问题则指求具有近似最大值(或近似最小代价)的解,而问题定义中还应规定所需近似程度。以下详述。

为了更准确,我们主要讨论希望取值最大化情形。为了更具表现性(或者其实是为了更灵活),允许解的取值也与实例本身有关。^③因此,对于(多项式界的)二元关系 R 及赋值函数 $f: \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbf{R}$,考虑使 f 取值最大的(R 的)解。即给定 x (满足 $R(x) \neq \emptyset$),求 $y \in R(x)$ 使得 $f(x,y) = v_x$,其中 v_x 为 $f(x,y')$ 对所有 $y' \in R(x)$ 求值的最大值。典型地, R 属于 $\mathcal{PC}$,而 f 为多项式时间可计算的。不失一般性,可以假设对任意 x ,存在某个多项式 l ,使得 $R(x) = \{0,1\}^{l(|x|)}$ (见习题2.8)。^④因此,上述优化问题可以重新描述为搜索问题:给定 x ,求 y 使得 $f(x,y) = v_x$,其中 $v_x = \max_{y' \in \{0,1\}^{l(|x|)}} \{f(x,y')\}$ 。

① 相反,对于受限的近似问题类存在着这样的分类。例如,参考文献[56]中对约束可满足问题(Constraint Satisfaction Problems)的近似版本)进行系统的分类。

② 在某种意义上,这种区别类似于上面提到的两种“近似”应用之间的区别。

③ 这个习惯只是为了方便:不失一般性,可将优化问题用其赋值函数表示,该函数只与对相应实例的每个解进行扩张后(即实例 x 的解 y 可被编码为一个对 (x,y) ,而对 x 的有效解进行这样处理之后得到的解集将包含所有形如 $(x,\cdot)$ 的对)得到的解有关。因此,上述习惯避免了对解的这种复杂编码。

④ 然而,此时函数 f 的取值必须同时依赖于实例及其解(即 $f(x,y)$ 的取值将不再与 x 无关)。

重点讨论相对近似问题。对某个缝隙函数 $g: \{0,1\}^* \rightarrow \{r \in \mathbf{R}, r \geq 1\}$, (最大化) 任务是求 y , 使得 $f(x, y) \geq v_x / g(x)$ 。事实上, 在某些情况下, 将近似因子定义为输入长度的函数 (即对某个 $g': \mathbf{N} \rightarrow \{r \in \mathbf{R}, r \geq 1\}$, 有 $g(x) = g'(|x|)$), 但是通常近似因子用关于输入的某个更精细的参数来表示 (如作为图中顶点个数的函数)。典型地, g 可在多项式时间内计算。

定义 10.1 (g -因子近似) 令 $f: \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbf{R}$, $l: \mathbf{N} \rightarrow \mathbf{N}$ 以及 $g: \{0,1\}^* \rightarrow \{r \in \mathbf{R}, r \geq 1\}$ 。

最大值版本: 对 f (关于 l) 求最大值时, g -因子近似是指搜索问题 R , 满足 $R(x) = \{y \in \{0,1\}^{l(|x|)} : f(x, y) \geq v_x / g(x)\}$, 其中 $v_x = \max_{y' \in \{0,1\}^{l(|x|)}} \{f(x, y')\}$ 。

最小值版本: 对 f (关于 l) 求最小值时, g -因子近似是指搜索问题 R , 满足 $R(x) = \{y \in \{0,1\}^{l(|x|)} : f(x, y) \leq g(x) \cdot c_x\}$, 其中 $c_x = \min_{y' \in \{0,1\}^{l(|x|)}} \{f(x, y')\}$ 。

注意: 对于大量的 NP-完全优化问题, 多项式时间算法能提供有意义的近似。10.1.1 节中将提到几个这样的例子。相反, 还存在许多其他的 NP-完全优化问题, 对其求近似在计算上等价于相应优化问题。10.1.1.2 节中也会提到几个这种例子, 另外还将介绍缝隙问题 (Gap Problem), 这是一个 (判定类型的) 承诺问题, 目的是为了了解释 (近似) 搜索问题的难度。

10.1.1.1 几个正面例子

首先介绍一个平凡的例子: 求图中的最大团。注意: 该问题有一种平凡的线性因子近似 (即给定图 $G = (V, E)$, 可以输出 V 中任意一个顶点, 作为 G 中最大团问题的 $|V|$ -因子近似)。

下面介绍一个著名的非平凡例子。

命题 10.2 (最小顶点覆盖的 2 倍近似) 存在多项式时间近似算法, 给定一个图 $G = (V, E)$, 可以输出一个顶点覆盖, 其规模最多为 G 中最小顶点覆盖规模的 2 倍。

注意: 由最小顶点覆盖的近似算法并不能得到其补问题 (即最大独立集) 的近似算法。这一现象与优化问题不同, 其中由一个问题的最优解 (如最小顶点覆盖) 可以得到其补问题的最优解 (最大独立集)。

证明概要: 观察到图的最大匹配集与顶点覆盖集之间存在某种联系。设 M 为图 $G = (V, E)$ 的最大匹配, 即 $M \subseteq E$ 是一个匹配, 但向其中增加一条任意边之后将不再是图的匹配。于是一方面, M 中所有顶点构成 G 的一个顶点覆盖, 另一方面, G 中每个顶点覆盖必须包含 M 中每条边的至少一个顶点。因此, 可以由最大匹配求出所需的顶点覆盖, 而最大匹配又可用贪心算法求出。□

另一个例子

巡回售货员问题 (Traveling Salesman Problem, TSP) 的实例由表示点对间距离的对称矩阵构成, 任务是求经过所有点的最短路径。通常对该问题没有合理且可行的近似解法 (见习题 10.1), 但是这里我们考虑两个特例, 其中的距离满足某种自然限制 (从而可以实现相当的近似)。

定理 10.3 (对特殊 TSP 的近似) 以下两个计算问题存在多项式时间算法:

(1) 当 TSP 实例中的距离满足三角不等式时, 求因子为 1.5 的近似。

(2) 对任意 $\varepsilon > 1$, 为欧几里得 TSP (Euclidean TSP, 即对某个常数 k (如 $k=2$), 实例中的点位于 k 维欧几里得空间中, 且两点间的距离为标准欧几里得范数) 求 $(1+\varepsilon)$ -因子近似。

习题 10.2 给出了问题 (1) 的一个较弱版本, 问题 (2) 的详细综述见参考文献[13]。注

意：定理 10.3 两个特例的差别：问题 (1) 是要对特定的常数因子提供多项式时间近似算法，问题 (2) 则是对任意常数因子提供近似算法。由此得到的结果被称为多项式时间近似方案 (Polynomial-time Approximation Scheme, PTAS)。

10.1.1.2 几个反例

还是从一个平凡例子出发。考虑求图中的最大团，给定一个图 $G=(V,E)$ ，求 G 中的 $(1+|V|^{-1})$ -因子近似最大团与求 G 中最大团难度相当。事实上，这个“结论”并没有什么实际意义。相反，根据 PCP 定理 (定理 9.16)，可以证明求图 $G=(V,E)$ 中 $|V|^{1-o(1)}$ -因子近似最大团与求 G 中最大团难度相同。这一结论可直接由近似问题的 NP-完全性 (见定理 10.5) 得到。

不可近似性结论在一类承诺问题中会更强，就是区分距离足够远实例的情形。这类承诺问题称为缝隙问题 (Gap Problems)，通常用两个有界函数 $g_1, g_2: \{0,1\}^* \rightarrow \mathbf{R}$ 来描述 (代替定义 10.1 中的缝隙函数)，且通常 g_1, g_2 是多项式时间可计算的。

定义 10.4 (f 近似的缝隙问题) 令函数 f 如定义 10.1 所述， $g_1, g_2: \{0,1\}^* \rightarrow \mathbf{R}$ 。

最大化版本：对 $g_1 \geq g_2$ ，求 f 最大值的 gap_{g_1, g_2} 问题是指区分 $\{x: v_x \geq g_1(x)\}$ 与 $\{x: v_x < g_2(x)\}$ ，其中 $v_x = \max_{y \in \{0,1\}^{f(|x|)}} \{f(x, y)\}$ 。

最小化版本：对 $g_1 \leq g_2$ ，求 f 最小值的 gap_{g_1, g_2} 问题是指区分 $\{x: c_x \leq g_1(x)\}$ 与 $\{x: c_x > g_2(x)\}$ ，其中 $c_x = \min_{y \in \{0,1\}^{f(|x|)}} \{f(x, y)\}$ 。

例如，求图中最大团的 gap_{g_1, g_2} 问题是指区分具有规模 $g_1(G)$ 的图 G 与不具有规模 $g_2(G)$ 的图 G 。此时，通常令 $g_i(G)$ 为关于图 $G=(V,E)$ 中顶点个数的函数，即 $g_i(G) = g_i'(|V|)$ 。事实上，用 $\omega(G)$ 表示图 G 中最大团的规模，而用 $\text{gapClique}_{L,s}$ 表示区分 $\{G=(V,E): \omega(G) \geq L(|V|)\}$ 与 $\{G=(V,E): \omega(G) < s(|V|)\}$ 的缝隙问题，其中 $L \geq s$ 。利用这些术语，我们重新表述 (并强化) 上述最大团问题的 $|V|^{1-o(1)}$ -因子近似。

定理 10.5 对某个 $L(N) = N^{1-o(1)}$ 及 $s(N) = N^{o(1)}$ ， $\text{gapClique}_{L,s}$ 问题为 NP-困难的。

证明定理 10.5 时需要定理 9.16 做较大修改，使其针对着均摊自由比特复杂度趋向于 0 的 PCP 系统 (见 9.3.4.1 节)。习题 10.3 给出了由定理 9.16 本身得到的一个更弱结论。

以下将证明，定理 10.5 类型的结论蕴含了相关近似问题的困难性，即判定一个缝隙问题的困难性蕴含了具有类似近似因子的搜索问题的困难性。

命题 10.6 设 f, g_1, g_2 如定义 10.4 所述，并假设这些函数可在多项式时间内计算，则对 f 求最大值 (或最小值) 的 gap_{g_1, g_2} 问题可以归约为求 f 最大值 (或最小值) 的 g_1/g_2 (或 g_2/g_1) - 因子近似问题。

注意：反方向的归约不一定存在 (即使是在底层的优化问题可以自归约的情况下)。事实上，这是本节讨论的问题与优化问题的另一个区别，在优化问题中，搜索问题可以归约为相关的判定问题。

证明概要：主要讨论最大化版本。对于输入 x ，用如下方法解决 gap_{g_1, g_2} 问题：发出询问 x ，得到应答 y ，当且仅当 $f(x, y) \geq g_2(x)$ 时，判定 x 的值不小于 $g_1(x)$ 。注意：只需对满足承诺的输入分析这个归约即可。因此，如果 $v_x \geq g_1(x)$ ，则预言必须返回一个解 y ，满足

$f(x, y) \geq v_x / (g_1(x) / g_2(x))$, 这蕴含了 $f(x, y) \geq g_2(x)$ 。另一方面, 如果 $v_x < g_2(x)$, 则对任意可能的解 y , 有 $f(x, y) \leq v_x < g_2(x)$ 。□

其他例子

考虑 $\text{gapVC}_{s,L}$, 即图中最小顶点覆盖的 gap_{g_s, g_L} 问题, 其中 s 和 L 为常数, 且对任意图 $G=(V, E)$, 有 $g_s(G)=s \cdot |V|$ (或 $g_L(G)=L \cdot |V|$)。命题 10.2 (通过命题 10.6) 蕴含了对任意常数 s , 问题 $\text{gapVC}_{s, 2s}$ 是多项式时间可解的。相反, 充分缩小缝隙两个阈值间的距离之后, 可以得到一个不可近似性结论。

定理 10.7 对某个常数 $s > 0$ 及 $L < 1$, 满足 $L > \frac{4}{3} \cdot s$ (如 $s=0.62, L=0.84$), $\text{gapVC}_{s,L}$ 问题是 NP-困难的。

定理 10.7 的证明基于对定理 9.16 一种复杂的修正。这里由定理 9.16 本身可再次得到一个较弱的结论 (见习题 10.4)。

对 PCP 定理 (定理 9.16) 的修改在建立不可近似性结论 (如定理 10.5 和定理 10.7) 方面起着关键作用。只需回顾定理 9.21 中的 PCP 定理本身等价于关于给定的 3-CNF 范式中可满足子句的最大数量的缝隙问题即可。具体地, $\text{gapSAT}_\varepsilon^3$ 定义为 (如定义 9.20) 一个缝隙问题, 针对着可满足的 3-CNF 范式与每个真值赋值至少不满足占比例 ε 的子句的 3-CNF 之间的缝隙。虽然定理 9.21 在定性断言之外并没有规定量化关系, 但是 (修正后) 与已知的最优 PCP 构造相结合, 可以得到可能界限的最佳值。

定理 10.8 对任意 $v < 1/8$, gapSAT_v^3 是 NP-困难问题。

另一方面, $\text{gapSAT}_{1/8}^3$ 是多项式时间可解的。

锋利的阈值

上述 (与 gapSAT_v^3) 相反的结论例证了近似因子的一个锋利阈值可由高效算法得到。另一个吸引人的例子针对着如下的最大化问题, 其中的实例为 $\text{GF}(2)$ 上的线性方程组, 而目标是求一种赋值, 能尽可能多地满足其中的方程。注意: 对于一种随机选择的赋值, 我们期望其能满足一半的方程。还要注意是否存在一种赋值能满足所有方程是易判定的。令 $\text{gapLin}_{L,s}$ 表示对两种方程组进行区分的问题, 即可以满足至少占 L 比例的方程组以及不能满足任意占 s 比例 (或更多) 的方程组。正如刚才所提示的, $\text{gapLin}_{L, 0.5}$ 是一个平凡问题 (对任意 $L \geq 0.5$), 而 $\text{gapLin}_{1,s}$ 是可行的 (对任意 $s < 1$)。相反, 对两个阈值进行修改, 使其远离两个相应极端, 则可得到一个 NP-困难的缝隙问题:

定理 10.9 对任意常数 $\varepsilon > 0$, 问题 $\text{gapLin}_{1-\varepsilon, 0.5+\varepsilon}$ 是 NP-困难的。

定理 10.9 的证明基于对定理 9.16 的修正。事实上, 相应的 PCP 系统 (NP 类的) 仅仅是定理 10.9 的一种重新阐述: 验证者利用对数数量的随机数发出 3 次询问, 并检查收到的应答是否满足线性条件。该验证者以至少 $1-\varepsilon$ 的概率接受任意一个 “是” 实例 (在给定向向适当证明的预言访问时), 并以至少 $0.5-\varepsilon$ 的概率拒绝任意一个 “否” 实例 (无论是否能访问预言)。习题 10.5 给出了由定理 9.16 本身得到的一个更弱结论。

缝隙的位置

定理 10.8 及定理 10.9 说明了关于 “缝隙” “位置” 的两种相反情形, 其中相应的承诺问题是困难的。注意: gapSAT 和 gapLin 都用两个阈值表示, 两个阈值分别限定了在 “是” 实

例和“否”实例中可满足的“本地”条件（即子句或等式）比例。在 gapSAT 中，较大的阈值（针对“是”实例）被设为 1，因此只有较小阈值（针对“否”实例）是一个自由参数。尽管如此，在 gapSAT 中仍证明了一个困难性结论，该结论针对较小阈值的最优值（见上述关于锋利阈值的讨论）。相反，在 gapLin 中，把较大阈值设为 1 将使缝隙问题可高效求解。因此， gapLin 的困难性建立在较大阈值的不同位置上。具体地，困难性（阈值比率的最优值）建立于将较大阈值设为 $1-\varepsilon$ 之上，对任意 $\varepsilon > 0$ 。

最后的说明

上述所有的不可近似性结论都涉及到对 NP 中优化问题进行弱化后得到的近似（或缝隙）问题（即该优化问题在计算上等价于 NP 中的判定问题，见 2.2.2 节）。在这些例子中，近似（或缝隙）的 NP -困难性蕴含了相应优化问题可归约为该近似（或缝隙）问题。换言之，这些例子中，对原始优化问题进行弱化，并不会得到更多的结果，因为弱化后的版本仍为困难问题。

10.1.2 判定或属性检测

当考虑实例间距离时，可以自然地得到对判定问题的一种弱化，其中的距离是指汉明距离（Hamming Distance）（即两个串对应比特不同的部分）。不严格地说，这种弱化（称为属性检测）是指区分来自某个预定集合 S 的输入和与该集合中任意输入距离“相对较远”的输入。这就产生了两种类型的承诺问题（关于任意预定集合 S （以及串间的汉明距离））：

(1) 关于固定相对距离的弱化判定问题：给定距离参数 δ ，考虑区分来自 S 与来自 $\Gamma_\delta(S)$ 中输入的问题，其中

$$\Gamma_\delta(S) \stackrel{\text{def}}{=} \left\{ x : \forall z \in S \cap \{0,1\}^{|x|}, \Delta(x,z) > \delta \cdot |x| \right\} \quad (10.1)$$

且 $\Delta(x_1 x_2 \cdots x_m, z_1 z_2 \cdots z_m) = |\{i : x_i \neq z_i\}|$ 表示串 $x = x_1 x_2 \cdots x_m$ 和 $z = z_1 z_2 \cdots z_m$ 中取值不同的位置数量。因此，这里考虑的承诺问题是对 S 中成员判定问题的一种约束（或一种特例）。

(2) 关于可变距离的弱化判定问题：问题的实例为对 (x, δ) ，其中 x 如类型 (1) 中定义， $\delta \in [0,1]$ 为（相对）距离参数。“是”实例是指 $x \in S$ 时的对 (x, δ) ，而 (x, δ) 为 $x \in \Gamma_\delta(S)$ 的一个“否”实例。

重点讨论类型 (1)，这似乎抓住了问题的本质，即是否这些弱化会降低原始判定问题的复杂度。对类型 2 的研究涉及另一个相对次要的问题，其中假设对上述问题的正面回答，即假设弱化后的形式比原始问题更容易，我们要考虑距离参数的减小（这意味着使弱化后的问题更“紧凑”，最终等价于原始问题）对复杂度会产生何种影响。

注意：对于大部分的 NP 完全问题而言，显然，存在可在多项式时间内解决的（第 (1) 类）弱化。实际上，这些算法在能直接访问输入时，运行于亚线性时间（具体是指多项式—对数时间）。10.1.2.2 节中将给出几个例子（但是，10.1.2.2 节中表明，这并非普遍现象）。在讨论这些例子之前，介绍几个重要定义。

10.1.2.1 定义

属性检测不仅与求解 NP -困难问题的弱化版本有关，而且还涉及到这些问题（以及 $\mathcal{P}$ 中的问题）是否能在亚线性时间内求解。当然，这些结论均假设算法可以直接访问输入（的表示）中的比特（见定义 10.10）。

定义 10.10（直接访问模型——习惯） 可以直接访问输入的算法是指主要输入位于一个

特殊的输入设备上, 算法将该设备作为预言来访问 (见 1.2.3.6 节)。另外, 还提供了输入长度以及保存于次要输入设备上的其他参数。算法的复杂度用主要输入的长度表示。

事实上, 5.2.4.2 节就提到这样一种模型, 但其中主要输入被视为预言而将次要输入当作真正输入。在直接访问模型下, 多项式一对数时间指算法所需时间为主要输入长度的多项式一对数, 其含义是算法的运行时间可表示为主要输入的二元表示长度的多项式。因此, 多项式一对数时间是一种特别高效的计算。我们将看到, 这种计算可以解各种 (属性检测) 问题。

定义 10.11 (S 的属性检测) 对任意给定的 $\delta > 0$, 区分 S 与 $\Gamma_\delta(S)$ 的承诺问题称为 S (关于 δ 的) 的属性检测。

回顾随机算法解承诺问题的定义, 如果算法以至少 $2/3$ 的概率接受每个“是”实例 (或拒绝每个“否”实例), 则称其解承诺问题。因此, 对 S 的一个 (随机化的) 属性检测以至少 $2/3$ 的概率接受 S 中的每个输入 (或拒绝 $\Gamma_\delta(S)$ 中的每个输入)。

表式问题

输入的具体表示在本节十分重要。这是由于表示方法对于距离的影响及可直接访问的机器依赖于输入的具体表示。以下详述这两个方面。

(1) 回顾在定义对象间距离时, 使用了汉明距离。显然, 此时表示方法的选择很关键, 不同表示方法产生不同距离。进一步地, 在这种情况下, 对象间距离在各种 (自然的) 表示方法中并不一致, 而这些方法在标准复杂性研究中被视为等价的。例如, 本书前面的章节中, 在讨论关于图的计算问题时, 并不考虑图是用邻接矩阵还是用依附列表表示。但是, 这两种表示方法可以得到截然不同的距离以及相应的属性检测问题 (见 10.1.2.2 节)。类似地, 填充 (以及其他平凡的句法习惯) 的使用将使问题更加复杂 (如大量填充会使所有对象看上去十分接近 (从而使对任意集合的属性检测成为一个平凡问题))。

(2) 由于我们重点讨论亚线性时间算法, 在算法中无法将输入由一种自然的表示转换为另一种。因此, 多项式时间算法中等价的表示方法在能直接访问对象表示的亚线性时间算法中可能并非等价。例如, 邻接询问和关联询问在较短时间内 (即顶点个数的亚线性时间内) 并不能相互模仿。

10.1.2.2 节中的例子将进一步澄清这两个方面。

承诺的重要作用

对于固定的常数 $\delta > 0$, 我们考虑区分 S 与 $\Gamma_\delta(S)$ 的承诺问题。承诺意指将忽略所有既不属于 S 也不远离 S (即不属于 $\Gamma_\delta(S)$) 的实例, 这对于亚线性算法来说是很关键的, 它使属性检测任务比相应的标准判定任务更容易 (见 10.1.2.2 节)。为证明这一点, 考虑集合 S 中的串大多数比特为 1 的情形。此时, 判定 S 的成员需要线性时间, 因为通过检测亚线性个位置, 并不能区分随机的含 $\lfloor n/2 \rfloor$ 个 1 的 n -比特串与含 $\lfloor n/2 \rfloor + 1$ 个 1 的 n -比特串 (即使在允许随机化和错误概率的情况下——见习题 10.8)。另一方面, 输入中含 1 的部分可用一个随机的多项式一对数时间算法近似求出 (从而得到 S 的属性检测器)。因此, 对于某些集合, 成员判定需要线性时间, 而属性检测可在多项式一对数时间内完成。

随机化的重要作用

针对上述例子, 我们指出, 在区分集合 S 与某个集合 (如 $\Gamma_{0.1}(S)$) 的任意亚线性时间算法中, 随机化是必要的。特别地, 一个亚线性时间的确定算法无法区分 1^n 与任意一个算法检测过的位置均为 1 的输入。一般来说, 对于输入 x , 一个 (亚线性时间) 确定算法总是读取 x

中相同位置处的比特, 因此, 如果存在一个串 z , 在这些位置处与 x 取值相同, 则算法无法区分 x 与 z 。

注意: 在两种情况下都可以证明算法时间复杂度的下界。这是因为这些下界在本质上具有信息论意义, 即这些下界实际上是指算法发出的询问数量。

10.1.2.2 检测图性质的两个模型

本小节讨论对于同构意义下封闭的图集进行属性检测的复杂度, 这种集合称为图的属性。鉴于属性检测中表示方法的重要性, 我们明确考虑图的两种标准表示 (见附录 G.1), 实际上得到检测图性质的两个不同模型。

(1) 邻接矩阵表示。将一个图 $G = ([N], E)$ (以某种冗余方式) 表示为 $N \times N$ 的布尔矩阵 $M_G = (m_{i,j})_{i,j \in [N]}$, 其中 $m_{i,h} = 1$ 当且仅当 $\{i, j\} \in E$ 。

(2) 有界的依附列表表示。对于固定的参数 d , 一个度数最多为 d 的图 $G = ([N], E)$ 被 (以某种冗余方式) 表示为一个映射 $\mu_G: [N] \times [d] \rightarrow [N] \cup \{\perp\}$, 使得当 v 为 u 的第 i 个邻居时有 $\mu_G(u, i) = v$, 而当 v 的邻居数小于 i 时, $\mu_G(u, i) = \perp$ 。

我们强调, 上述表示方法均可以确定两个图之间的距离以及由算法发出的询问类型。以下将说明, 由于表示方法的不同, 将使相应的属性检测问题具有截然不同的复杂度。

定理 10.12 (邻接矩阵表示下的属性检测) 对任意固定的 $\delta > 0$ 以及以下每个集合, 存在一个多项式—对数时间随机算法, 可以 (关于 δ) 解相应的属性检测问题。

- 对任意固定的 $k \geq 2$, k -着色图的集合。
- 对任意固定的 $\rho > 0$, 包含一个 ρ -规模团 (或独立集) 的图的集合。
- 对任意固定的 $\rho > 0$, 包含至少有 $\rho \cdot N^2$ 条边的切割^① (Cut) 的 N -顶点图集合。
- 对任意固定的 $\rho > 0$, 包含至多有 $\rho \cdot N^2$ 条边的对分^② (Bisection) 的 N -顶点图集合。

相比之下, 对某个 $\delta > 0$, $\mathcal{NP}$ 中存在一个图属性, 其中 (关于 δ) 的属性检测需要线性时间。

检测算法 (见定理 10.12) 利用常数次询问, 该常数是常量 $1/\delta$ 的多项式。相反, 相应集合的精确判定过程需要线性数量的询问。上述算法的运行时间中隐藏了一个常量, 其大小是询问复杂度的指数 (除了图的 2-着色属性, 其中隐藏的常数是 $1/\delta$ 的多项式)。注意: 这种依赖关系是必要的, 因为令 $\delta = 1/N^2$ 即可再次得到原始的 (未弱化的) 判定问题 (这些问题, 除了 2-着色图以外, 都是 NP-完全问题)。至于下界 (见定理 10.12), 我们认为证明了下界的图属性并非正常现象。10.1.2.1 节中指出, 时间复杂度的下界可由询问复杂度下界得了出。

定理 10.12 阐明了通过常数次询问检测的图属性以及需要线性次询问才能检测的图属性之间的巨大差别。已知一种图属性的组合特征, 其中的属性检测 (在邻接矩阵表示下) 可利用常数次询问完成。^③注意: 这里的常数可能与 δ 相关 (并且事实上在某些情况下, 是比 $1/\delta$ 的指数增加得更快一个函数)。例如, 对于无三角形 (Triangle-free) 的图集进行属性检测时使用的询问次数只取决于 δ , 但是已知这个数量一定比 $1/\delta$ 的任意多项式增长得更快。

回到定理 10.12, 注意: 从有关最大切割和最小对分图集的属性检测问题的结论出发, 可

① 图 $G = ([N], E)$ 的一个切割是对顶点集的一种划分 (S_1, S_2) (即 $S_1 \cup S_2 = [N]$ 且 $S_1 \cap S_2 = \emptyset$), 切割中的边恰好有一个端点在 S_1 中。图的对分 (Bisection) 是指两部分顶点数相同的切割。

② Alon 等关于规范 (Regular) 切割 (由 Szemerédi 引入) 的这一结论超出了本章的讨论范围。

以得到具有加法误差项（为 δN^2 ）的近似算法。密集图（Dense Graph）（即含 $\Omega(N^2)$ 条边的 N -顶点图）的标准近似问题（如定义 10.1）可以有一个常数因子近似。即对任意常数 $c > 1$ ，在密集图中使切割规模最大化（或使对分规模最小化）的问题，可以得到一个 c -因子近似。另一方面，由关于团的结论可以得到最大团问题的一个对偶近似，即在具有给定规模的最密集诱导子图中，可以近似求出被去掉边的最小数量。

事实上，定理 10.12 只对密集图有意义。通常在任意以邻接矩阵表示的图属性中该结论均成立。^①还要注意在邻接矩阵表示下，针对任意满足 $\Gamma_{o(1)}(S) = \emptyset$ 的图属性 S （如连通图集合、哈密尔顿图集合等）的属性检测都是平凡的。

现在讨论图的有限依附列表表示，这只对度数有限的图有意义。最大切割、最小对分及团（如定理 10.12）在这种表示下是平凡的，但图的连通性是非平凡的，并且对于二部图属的属性检测的复杂度也有较大改变。

定理 10.13（有限依附列表表示下的属性检测） 以下断言针对着图的长为 d 的依附列表表示。

- 对任意给定的 d 及 $\delta > 0$ ，存在一个多项式-对数时间随机算法，可求解度数不超过 d 的连通图属的属性检测问题。
- 对任意给定的 d 及 $\delta > 0$ ，存在一个亚线性时间随机算法，可求解度数不超过 d 的二部图属的属性检测问题。具体地，对于输入的 N -顶点图，算法的运行时间为 $\tilde{O}(\sqrt{N})$ 。
- 对任意给定的 $d \geq 3$ 及某个 $\delta > 0$ ， N -顶点（3-正则）二部图属的属性检测需要 $\Omega(N)$ 次询问。
- 对某个给定的 d 及 $\delta > 0$ ，度数不超过 d 的 N -顶点、3-着色图的属性检测需要 $\Omega(N)$ 次询问。

定理 10.13 中算法的运行时间里隐藏了一个常数，取值为 $1/\delta$ 的多项式。寻找图属性检测器的复杂度特征（在有限依附列表表示下），是一个有趣的公开问题。

距离与表示方法分离的情形：

至此我们只考虑了用不同方法表示时的汉明距离。这使表示方法的选择格外重要（比复杂性理论的一般情形更重要）。相反，人们还会很自然地考虑图的距离概念能否独立于具体表示方法。例如，图 $G_1 = (V_1, E_1)$ 与 $G_2 = (V_2, E_2)$ 间的距离定义为 E_1 和同构于 G_2 的图边集间的最小对称差。此时，相对距离可以定义为距离除以 $E_1 + E_2$ （或 $\max(|E_1|, |E_2|)$ ）。

10.1.2.3 在图属性之外

除了图论之外，属性检测还出现在许多计算问题中。事实上，这类问题首先出现在代数领域，其中的实例（视为检测算法的输入）为函数，而相应属性为代数函数集。一个典型例子是低次多项式：即某个有限域 $\text{GF}(q)$ 上次数为 d 的 m 变量多项式，其中 m 、 d 和 q 取决于输入长度（或满足某种关系，如 $q = d^3 = m^6$ ）。注意：此时，输入为 $\text{GF}(q)$ 上 m -变量函数的

① 以下将证明，在这种模型中，非密集图的属性检测是平凡的。具体地，给定距离参数 δ ，称一个 N -顶点图是非密集的，如果其中有少于 $(\delta/2) \cdot \binom{N}{2}$ 条边。关键在于，对于非密集图，任意集合 S 的属性检测问题是平凡的，因为当且仅当 S 中包含某个非密集（ N -顶点）图时，即可接受任意的非密集（ N -顶点）图。显然，当 S 中不含非密集图时，这一判断是正确的，而当 S 中包含非密集图时，该判定也可接受，因为此时每个非密集图都与 S 是“ δ -接近的”（即不属于 $\Gamma_\delta(S)$ ）。

(“完全”或“明确”)描述,这意味着其长度为 $q^m \cdot \log_2 q$ 。将函数作为问题实例暗示了一种自然的距离度量方法(即函数取值不同的变量部分),并且存在自然的方法来访问距离(即对于选择的变量,询问函数的取值即可)。

我们已经用不同的术语体系讨论了这些计算问题,见 9.3.2.2 节和 9.3.2.1 节。特别地,在 9.3.2.1 节中讨论了一种特例即线性布尔函数(次数为 1 且 $q=2$),而在 9.3.2.2 节中,令 $q = \text{poly}(d)$, $m = d/\log d$ (其中 d 为函数次数的上限)。

其他涉及属性检测的计算问题来自于几何学(如聚类问题)、形式语言(如正则集的成员检测)、编码理论(见附录 E.1.3)、概率论(如检测分布的等价性)以及组合数学(如单调函数)等领域。在 10.1.2.2 节的结尾将讨论,将距离测试与对象的表示方法(即对问题实例的访问方式)相分离是一种自然的做法。在实现时,需引入独立于实例间距离的表示方法。

10.2 平均情况复杂性

教学注释:我们将平均情况复杂性看作是“平均”(或典型)实例的性能,而非随机实例的平均性能。在 10.2.1.1 节中这是合理的。因此,将以下理论称为典型情况复杂性也许更合理。然而,由于历史原因,平均情况复杂性度的名称仍保留下来。

目前为止,我们的方法(包含在 10.1 节)都称为最坏情况复杂性,因为它涉及算法对于所有合法实例的性能(从而也包括最坏实例的性能)。也就是说,计算问题的定义中包括一个实例集,要求对该集合中每个实例能保证性能。相反,平均情况复杂性允许忽略极少数实例,其中被忽略实例取决于潜在的解决者,而非问题描述。

这里要给出几个说明。首先,正如刚刚提到,计算问题(特别是有承诺的问题)的最坏情况复杂度的标准陈述中可能也忽略了一些实例(如那些被认为是不可接受的或违背承诺的实例),但这些实例取决于问题的描述。相反,在平均情况复杂度下,被忽略的输入在本质意义上并非完全不可接受(并且肯定不能在问题描述中被识别)。在某个具体算法解该问题时,将这些实例视为例外是合理的,即这些特殊实例由对算法的分析而确定。当然,这些实例应该数量极少(以可忽略的概率出现)。

上一段中最后一句话引出几个问题。最重要的是,必须规定可接受实例集中的分布。实际我们将考虑一类新的计算问题,每个这种问题由一个标准计算问题和实例集上的一种分布构成。因而,在平均情况复杂度理论中应该考虑采用何种分布的问题。10.2.1.1 节中将讨论这个问题,并给出一些相关定义。

首先阐明研究计算问题的平均情况复杂度一个相当直接的动机:那就是在实际应用中,人们会满足于一个能对几乎所有实例快速求解的算法。也就是说,人们可能会容忍以可忽略概率出现的错误,其中的概率来自于实例集上的分布。平均情况复杂度的研究目标是如何从这种弱化中受益,并区分能有益的情形与无益情形。此时,一个关键方面是对正常的算法应用中出现的(实例的)分布类型建立好的模型。

我们从另一个不同角度考虑上述动机: $P \neq NP$ (或更确切地 $NP \not\subseteq BPP$) 的假设只认为不可解性是 NP 中某些问题的某些实例的特征。这些不可解的实例可能十分稀少且为“病态”的。平均情况复杂性则研究不可解性是否也是“典型”实例的特征(即是否不可解的实例会关于某种简单分布以不可忽略的概率出现)。当然,后一个问题是否有意义依赖于对分布

类型进行限制，只允许简单的分布（而非“病态”分布）。我们将考虑两种分布（分别见于 10.2.1.1 节和 10.2.2.2 节），并证明如果不可解性出现于一个更大的类中，则也会出现于限制条件更严格的类中（见定理 10.26）。

$P \neq NP$ 的平均情况版本

事实上，一个根本问题是是否每个自然的计算问题在限定到典型实例时都能高效求解。本节提出的猜想是：对于良好选择的定义，答案是否定的；即猜想 NP 类的“分布式版本”不包含于 P 类的平均情况（或典型情况）版本。这意味着某些 NP 问题不仅在最坏情况下是困难的，而且是“典型困难”的（即根据某种简单分布提取的典型实例是困难的）。这暗示了困难实例可能会出现于正常的算法应用中（而不仅是密码学这种有意生成困难实例的应用（或其他“对抗性的”应用））。^①

上述猜想引入 NP -完全性在平均情况下的研究，这是本节的主要内容。事实上，本节相当于第 2 章在平均情况下的对应。特别地，在假设“典型困难”的分布问题存在时，这种（平均情况）理论可以识别“典型困难”的分布问题。如果上述猜想成立，则针对这种完全的（分布）问题，可以认为寻找高效求解典型实例的算法是徒劳的。

内容组织

大部分内容涉及平均情况复杂性的通用理论中出现的概念问题。10.2.1.1 节中给出了相关定义。10.2.1.2 节中证明了在平均情况复杂性意义上，存在着“ NP -完全”的分布问题。特别地，我们将说明对任意 NP -完全判定问题，如何得到这样一个分布式的版本。10.2.1.3 节中对随机算法进行扩展。10.2.2 节介绍其他一些研究分支。

10.2.1 基础理论

本节介绍平均情况复杂性的基础理论，而将重要的研究分支介绍放到后面的 10.2.2 节。基础理论包括主要定义以及如何识别平均情况计算问题类中的完全问题。

10.2.1.1 定义

平均情况复杂性的理论可能比乍乍看起来更微妙。除了在弱化的定义中遇到的概念性困惑之外，难点还来自于标准概率分析和复杂性理论习惯的“界面”。这在可行的平均情况计算类的定义中最为显著。参考最坏情况下的复杂性理论，平均情况复杂性的理论应解决以下几方面问题。

（1）构造通用的框架。我们将主要考虑分布问题，这是指标准计算问题（见 1.2.2 节）辅之以相关实例的分布。

（2）识别可行的（分布）问题类。在为一些问题类，如 P 类，寻找平均情况下的对应时，我们应抛弃头脑中首先出现的定义（即“平均多项式时间”的天真概念），而是考虑几个相关的代替者，并采用其中之一。

（3）识别相关（分布）问题类。为 NP 类寻找平均情况下的对应物时，应同时避免两个极端，一种是允许任意的分布（这将使平均情况困难性退化为最坏情况困难性），另一种恰好相反，局限于某种分布如均匀分布。

（4）充分发展（分布）问题间归约的概念。与最坏情况困难性理论相同，这一概念应当

^① 我们强调本节（正常的算法应用）与密码学的两个不同之处。首先，在本节中，提到典型困难的问题时，输入分布的简单性十分关键：分布越简单，则困难断言越强大。然而这种考虑对于密码学而言是不足轻重的。另一方面（见 7.1.1 节一开始或 10.2.2.2 节最后的讨论），密码学应用要求能高效地生成困难实例及相应的解。

保持其可解性（在当前分布下）。

现在转到对上述问题的实际处理方法。

第一步：定义分布问题

重点讨论判定问题，分布问题定义为由一个判定问题和一个概率空间组成的对。^①为简单起见，这里概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 是一系列随机变量，其中 X_n 属于 $\{0,1\}^n$ 。因此，分布问题 $(S, \{X_n\}_{n \in \mathbb{N}})$ 由集合 S 的成员判定问题及概率空间 $\{X_n\}_{n \in \mathbb{N}}$ 组成（对搜索问题的处理是类似的，见 10.2.2.1 节）。将均匀概率空间记作 $U = \{U_n\}_{n \in \mathbb{N}}$ ，即 U_n 在 $\{0,1\}^n$ 上均匀分布。

第二步：识别可行的问题类

问题 $(S, \{X_n\}_{n \in \mathbb{N}})$ 被定义为（平均情况下）可行的，如果存在解 S 的算法 A ，其对 X_n 的平均运行时间是 n 的多项式（即存在一个多项式 p ，使得 $E[t_A(X_n)] \leq p(n)$ ，其中 $t_A(x)$ 表示 A 对于输入 x 的运行时间）。这个定义易受计算模型影响，且对于算法的合成不封闭。两个缺陷都可归因于一个事实，即 t_A 通常可能是关于 $\{X_n\}_{n \in \mathbb{N}}$ 的多项式，而 t_A^2 不是（如考虑当 $x' = x^n$ 时， $t_A(x'x^n) = 2^{|x|}$ ，否则 $t_A(x'x^n) = |x'x^n|^2$ ，伴随着 $\{0,1\}^n$ 上的均匀分布）。由此我们断定算法的平均运行时间不是一个健壮的概念。我们还质疑上述定义中条件的合理性，并认为算法的典型运行时间（以下给出定义）的概念更自然。因此，认为一个算法是可行的，如果其运行时间典型地为多项式。^②

称算法 A 对于 $X = \{X_n\}_{n \in \mathbb{N}}$ 是典型多项式时间的，如果存在多项式 p ，使得 A 对 X_n 的运算超过 $p(n)$ 步的概率可忽略（即对任意多项式 q 及所有足够大的 n ，有 $\Pr[t_A(X_n) > p(n)] < 1/q(n)$ ）。问题在于“非典型”情况是什么？以下给出两种可能的定义。

(1) 一种更简单的定义称问题 $(S, \{X_n\}_{n \in \mathbb{N}})$ 为（典型）可行的，如果存在解 S 的算法 A 使得 A 对于 $X = \{X_n\}_{n \in \mathbb{N}}$ 为典型多项式时间的。这要求 A 对 S 的每个实例均能正确求解，比前面的动机讨论要求要高（事实上，如果动机是忽略不常见情况，则也可以将其全部忽略而非只忽略一部分（即只忽略对运行时间的影响））。

(2) 另一种定义与动机讨论中一致，称 (S, X) 为（典型）可行的，如果存在多项式时间算法 A ，可以在多项式时间内典型地对 X 上的 S 求解，即存在一个多项式 p ，使得对于输入 X_n ，算法 A 出错或者运算时间多于 $p(n)$ 步的概率可以忽略。这一定义完全忽略了非典型实例。事实上，不失一般性，此时，可以假设 A 总是运行于多项式时间内（见习题 10.11），但是这里不能这样假设（这是为了便于将第一种定义当作第二种的特殊情形）。

我们强调上述两种定义实际上都定义了典型可行性，而非平均情况可行性。为了阐述两者的区别，考虑判定一个均匀选择的 $(n-顶点)$ 图是否能 3-着色的分布问题。直观上，这一

① 我们指出，这种定义并非显然。Levin^[154]（见参考文献[89]中的讨论）提倡使用定义于所有串集合上的单个概率分布。他的根据是这将使理论更少地依赖于表示方法。当时我们认可这种观点（见参考文献[89]），但后来发现参考文献[89]中的讨论依赖于表示的效果是合理的。进一步地，参考文献[89, 154]中的另一种定义偶尔会不那么自然，从而使一些读者困惑。

② 另一种定义由 Levin 给出^[154]（见参考文献[85]中的讨论），将一个算法视为可行的（关于 $X = \{X_n\}_{n \in \mathbb{N}}$ ），如果其运行时间是平均意义上为线性的函数（关于 X ）的多项式，即要求存在一个多项式 p 以及函数 $l: \{0,1\}^* \rightarrow \mathbb{N}$ ，使得对任意 x 有 $l(x) \leq p(l(x))$ 以及 $E[l(X_n)] = O(n)$ 。这个定义是健壮的（不存在上述缺陷）并且可以论证与平凡定义（即 $E[t_A(X_n)] \leq \text{poly}(n)$ ）一样“自然”。

问题是“典型平凡的”(关于均匀分布),^①因为算法可以总是回答“否”,而这样做的出错概率会呈指数递减。事实上,这种平凡算法可以被第二种定义接受,但并不被第一种接受。根据上述讨论,我们采用第二种定义。

定义 10.14 (tpcP 类) 称算法 A 在多项式时间内典型地解 $(S, \{X_n\}_{n \in \mathbb{N}})$, 如果存在一个多项式 p , 使得对于输入 X_n , A 或者出错或者运行超过 $p(n)$ 步的概率可以忽略。^②我们将可以典型地在多项式时间内求解的分布问题类记作 tpcP 类。

显然, 对于每个 $S \in \mathcal{P}$ 以及所有概率空间 X , 有 $(S, X) \in \text{tpcP}$ 。然而, tpcP 中也包含这样一些分布问题 (S, X) , 其中 $S \notin \mathcal{P}$ (见习题 10.12 和习题 10.13)。平均情况复杂性理论中一个基本问题是 NP 的自然分布版本是否属于 tpcP。因此, 我们要研究如何识别这种版本。

第三步: 识别相关问题类

为了识别 NP 的合理分布版本, 要注意避免两种极端选择。一方面, 必须限制允许的分布集, 从而防止平均情况困难性退化为最坏情况困难性(通过选择“最坏情况”实例上的病态分布来做到这一点)。另一方面, 应该允许各类自然分布, 而不仅是均匀分布。^③回顾我们的目标是要处理应用中所有可能出现的输入分布, 因此局限于均匀分布是不合理的。然而, 按理说, 实际中的分布都“相对简单”, 所以需要识别简单的分布类。这种概念(简单分布)是以下定义的基础, 10.2.2.2 节中将介绍一个更弱化的概念。

定义 10.15 (distNP 类) 称概率空间 $X = \{X_n\}_{n \in \mathbb{N}}$ 是简单的, 如果存在多项式时间算法, 对任意输入 $x \in \{0, 1\}^*$, 能输出 $\Pr[X_{|x|} \leq x]$, 其中不等式针对着字符串的标准词典序。用 distNP 表示包含 NP 中判定问题伴以简单概率空间的分布问题类。

注意: 均匀的概率空间是简单的, 但许多其他“简单”概率空间也是简单的。实际上, 一种有意义的做法是放宽定义要求, 使算法只输出 $\Pr[X_{|x|} \leq x]$ 的近似值, 如在近似因子 $1 \pm 2^{-2|x|}$ 之内。注意: 定义 10.15 用计算术语解释简单性, 具体地, 将其定义为关于概率分布的基础问题作出回答的可行性(如确定单个 $(n$ -比特长)字符串, 甚至字符串的一部分的概率)。这种简单性条件与通过单调映射在多项式时间内采样的概念极为接近(见习题 10.14)。

教学注释: 以下两段试图回答针对定义 10.15 的疑问, 可延迟阅读。

将简单分布标识为有意义的分布类, 比本书中其他标识更令人困惑。尽管如此, 我们相信拒绝上述两种极端(即或者允许所有分布, 或者只允许均匀分布)是完全合理的。然而读者可能会疑惑是否能在“通用性”与“简单性”(从直观上)之间取得平衡。一个具体考虑是可能对分布类进行了太多的限制。下面简要回答这个问题。

多项式时间可采样的概率空间(见 10.2.2.2 节)是一个更直观和非常健壮的分布类, 它似乎包含了实际应用中出现的所有分布。幸运的是, 结合 10.2.1.2 节与 10.2.2.2 节中的结论, 似乎支持定义 10.15 中的选择。具体地, 注意: 对分布类型的扩展将减弱相应分布类 NP 问

① 相反, 测试一个给定图是否可 3-着色, 对于其他分布似乎是“典型困难”的(见定理 10.19 或习题 10.27)。当然, 在后一种分布中, “是”实例和“否”实例均以显著概率出现。

② 称一个函数 $\mu: \mathbb{N} \rightarrow \mathbb{N}$ 为可忽略的, 如果对任意正多项式 q 及所有足够大的 n , 有 $\mu(n) < 1/q(n)$ 。对于谓词 $x \in S$, 如果 $A(x)$ 不同于 x 的指标值(Indicator Value), 则称 A 关于 x 出错。

③ 只关注均匀分布会产生一种天真的想法, 即认为该分布是应用中涉及的唯一分布, 并被这种想法误导。相反, 我们认为, 对于大多数正常应用, 实例集上的均匀分布毫无意义。

题中包含不可解问题的猜想。另一方面,如果某个特定分布问题属于一个对应于受限允许分布的更小类,则该问题不可解的结论将更吸引人。现在,将 10.2.1.2 节与 10.2.2.2 节中的结论相结合,可以断言,在更大的多项式时间可采样概率空间中的猜想,蕴含了更简单概率空间(属于更小的类)中的结论。因此,目前建立于简单概率空间中的猜想和结论具有过渡性质。

事实上,本节的主要问题是 distNP 是否包含于 tpcP 中。对该问题的肯定回答(特别是扩展到可采样概率空间时)将认定 P-vs-NP 问题只有很小的实用价值。然而,日常经验以及许多研究工作表明某些 NP 问题不仅在最坏情况下是困难的,而且更是“典型困难”的。这就引出一个猜想,即 distNP 不包含于 tpcP 之中。

当然,这个猜想蕴含了 $P \neq \text{NP}$, 从而不指望看到其证明。具体而言,我们不应期望证明 distNP 中某个特定问题不属于 tpcP 。而有希望见到的是“ distNP -完全”问题,即在 distNP 中存在一些问题不属于 tpcP , 除非整个 distNP 类都包含于 tpcP 中。以下利用一个充分归约来阐述这种可能性。

第四步: 定义(分布)问题间的归约

直观上,归约必须保持平均情况下的可解性。因此,除了归约的标准条件(即归约可以高效计算且得到正确结果)之外,还要求归约“服从”相应分布问题的概率分布。具体地,归约不应将第一个问题(“起点”)的常见实例映射为第二个问题(“目标”)的罕见实例,否则,由解第二个问题的典型多项式时间算法将无法得到解第一个问题的类似算法。以下是对 Cook-归约(即通用的多项式时间归约)的充分模拟,而对 Karp-归约(多到一归约)的模拟是一种特例。

教学注释: 建议在课堂上只介绍多到一归约,这足以证明定理 10.17。

定义 10.16(分布问题间的归约) 称预言机 M 将分布问题 (S, X) 归约为分布问题 (T, Y) , 如果以下 3 个条件成立。

(1) 高效性。机器 M 运行于多项式时间。^①

(2) 正确性。对任意 $x \in \{0, 1\}^*$, 当且仅当 $x \in S$ 时, $M^T(x) = 1$, 其中 $M^T(x)$ 表示 M 在可以访问 T 时对于输入 x 的输出。

(3) 支配性。^②对于输入 X_n 及 T 的预言访问, 机器 M 的询问 y 具有上界 $\text{poly}(|y|) \cdot \Pr[Y_{|y|} = y]$ 。即存在一个多项式 p , 使得对任意 $y \in \{0, 1\}^*$ 及任意 $n \in \mathbf{N}$, 有

$$\Pr[Q(X_n) \ni y] \leq p(|y|) \cdot \Pr[Y_{|y|} = y] \quad (10.2)$$

式中: $Q(x)$ 表示 M 对于输入 x 和预言访问 T 发出的询问集合。

另外,要求归约中的询问不应太短,即存在多项式 p' , 使得当 $y \in Q(x)$ 时有 $p'(|y|) \geq |x|$ 。

式(10.2)左边是对于服从分布 X_n 的输入, 归约发出询问 y 的概率。要求这一概率不能

① 实际上,可以放宽该要求,只要求 M 关于 X 是典型多项式时间的即可。对正确性条件也可作类似放宽。

② 我们利用多到一归约的特殊情形(即 $M^T(x) = 1$ 当且仅当 $f(x) \in T$, 其中 f 为多项式时间可计算的函数)来详细说明式(10.2)的含义: 此时,用 $\Pr[f(X_n) = y]$ 代替 $\Pr[Q(X_n) \ni y]$, 即式(10.2)被简化为 $\Pr[f(X_n) = y] \leq p(|y|) \cdot \Pr[Y_{|y|} = y]$ 。事实上,这一条件对任意不属于 f 的像的 y 而言是无意义的。

超过 y 出现于分布 $Y_{|y|}$ 中的概率与 $|y|$ 的多项式之乘积。此时, 称式 (10.2) 的左边受 $\Pr[Y_{|y|} = y]$ 支配。

事实上, 支配性条件是定义 10.16 中唯一对最坏情况下归约进行扩展并涉及分布环境的条件。支配性条件并不坚持 $Q(X)$ 诱导出的分布等于 Y , 但是允许某种松弛, 即存在一个上界, 从而可以维持典型可解性 (见习题 10.15)。①

注意: 第 7 章、第 8 章中大量使用的可约性论据 (见 7.1.2 中的讨论) 实际上就是定义 10.16 中的归约 (只是针对的计算任务类型不同)。

10.2.1.2 完全问题

回顾 $\text{dist}\mathcal{NP}$ 不包含于 $\text{tpc}\mathcal{P}$ 中的猜想, 这又反过来强化了 $\mathcal{P} \neq \mathcal{NP}$ 的猜想 (使一个典型现象而非最坏情况下的现象成为不可行)。在没有希望证明 $\text{dist}\mathcal{NP}$ 不包含于 $\text{tpc}\mathcal{P}$ 中时, 我们转而研究关于该猜想的完全问题。具体地, 称一个分布问题 (S, X) 是 $\text{dist}\mathcal{NP}$ -完全的, 如果 $(S, X) \in \text{dist}\mathcal{NP}$ 且任意 $(S', X') \in \text{dist}\mathcal{NP}$ 可归约到 (S, X) (在定义 10.16 的归约下)。

证明 NP-完全问题的存在性相当容易, 许多自然问题都是 NP-完全的。相反, 在本节要证明完全性结论则十分困难。鉴于本节考虑的是受限类型的归约, 因此这一点并不奇怪。其中的约束条件 (体现于支配性条件中) 要求不应将一个问题的“典型”实例映射为另一问题的“非典型”实例。然而, 称标准 Karp-归约 (用于证明 NP-完全性结论时) 将一个问题的“典型”实例映射为第二个问题的“奇异”实例是合理的。因此, 认为本小节研究的归约不会犯这种“错误”。②

定理 10.17 (dist $\mathcal{NP}$ -完全性) $\text{dist}\mathcal{NP}$ 中包含着分布问题 (T, Y) , 满足 $\text{dist}\mathcal{NP}$ 中每个分布问题都可归约 (由定义 10.16) 为 (T, Y) 。进一步地, 这种归约为多到一映射。

证明: 首先引入这样一个 (分布) 问题, 它是判定问题 S_u (用于定理 2.19 的证明中) 的分布式版本。如果存在 $y \in \bigcup_{i \leq t} \{0, 1\}^i$ 使得 M 在 t 步内接受输入对 (x, y) , 则 S_u 中包含实例 $\langle M, x, 1' \rangle$ 。将 S_u 与“半均匀”的概率空间 U' 相关联, U' 为实例 $\langle M, x, 1' \rangle$ 分配的概率与 $2^{-(|M|+|x|)}$ 成正比。具体地, 对每个 $\langle M, x, 1' \rangle$, 有

$$\Pr[U'_n = \langle M, x, 1' \rangle] = \frac{2^{-(|M|+|x|)}}{\binom{n}{2}} \quad (10.3)$$

式中: $n = \left| \left\langle M, x, 1' \right\rangle \right| \stackrel{\text{def}}{=} |M| + |x| + t$ 。注意: 在适当的编码方式下, U' 确实是简单的。③

读者易验证在将 $\mathcal{NP}$ 中任意集合归约为 S_u (见定理 2.19 的证明) 时所使用的通用归约,

① 我们强调支配性的概念与统计 (或计算) 不可区分的概念不可比。一方面, 被支配是一种本地需求 (即对两个分布进行逐点比较), 而不可区分性是全局性需求 (它允许个别例外)。另一方面, 支配性不要求近似相等的值, 而是在一个方向的某个比率。支配性不是对称的。我们指出, 关于支配性有一个更宽松的概念, 其中允许个别例外, 这样也足以保持典型的可行性。

② 最后这个断言是有争议的。然而, 在定理 10.17 的证明中它似乎完全合理, 对定理 10.19 的证明会有不同看法。

③ 例如, 可将 $\langle M, x, 1' \rangle$ 编码为串 $\sigma_1 \sigma_2 \cdots \sigma_k 0 1 \tau_1 \tau_2 \cdots \tau_l 0 1'$, 其中 $M = \sigma_1 \sigma_2 \cdots \sigma_k \in \{0, 1\}^k$, $x = \tau_1 \tau_2 \cdots \tau_l \in \{0, 1\}^l$, 则 $\Pr[U'_n = \langle M, x, 1' \rangle] \leq \frac{\binom{n}{k+l}}{\binom{n}{2}} = \frac{2^{-(|M|+|x|)}}{\binom{n}{2}} \cdot \left[\left| \left\{ M' \in \{0, 1\}^{|M|} : M' < M \right\} \right| + 2^{-(|M|+|x|)} \cdot \left| \left\{ x' \in \{0, 1\}^{|x|} : x' \leq x \right\} \right| \right]$

式中: $i_{k,l,t}$ 为 $\{k, k+t\}$ 在 $[k+l+t]$ 的所有 2-子集中的序号。

无法将 distNP 归约为 (S_u, U') 。具体地, 在某些情况下 (见下一段落), 这些归约不满足支配性条件。事实上, 难点在于我们必须将所有的 distNP 问题 (即由判定问题和简单分布构成的对) 归约为简单的分布问题 (即 (S_u, U'))。相反, 考虑由上述归约诱导出的分布, 可以得到 S_u 的许多分布式版本, 而且这些分布完全不同 (且不一定被同一个分布支配)。

更深入地研究一下上述通用归约 (S 到 S_u) 应用于任意 $(S, X) \in \text{distNP}$ 的情形。该归约将实例 x 映射为三元组 $(M_S, x, 1^{p_S(|x|)})$, 其中 M_S 为识别 S 中成员的机器 (利用足够的 NP-证据), p_S 是一个充分的多项式。但是 x 可能概率相对较大 (即可能有 $\Pr[X_{|x|} = x] \gg 2^{-|x|}$), 而 $(M_S, x, 1^{p_S(|x|)})$ 有“均匀的”概率 (即 $(M_S, x, 1^{p_S(|x|)})$ 在 U' 中的概率小于 $2^{-|x|}$)。这就不符合支配性条件 (见习题 10.18), 因此, 需要使用其他归约。

这里归约的关键在于利用均匀分布下具有某种概率的串对任意简单分布下具有类似概率的串进行 (高效) 编码。这意味着编码时要对在原始分布下概率较大的串进行压缩。具体而言, 将 x (来自于概率空间 $\{X_n\}_{n \in \mathbb{N}}$) 映射为码字 x' , 其长度上限为 $1/\Pr[X_{|x|} = x]$ 的对数, 并确保 $\Pr[X_{|x|} = x] = O(2^{-|x'|})$ 。相应地, 归约中将 x 映射为三元组 $(M_{S,X}, x', 1^{p'(|x|)})$, 其中 $|x'| < O(1) + \log_2(1/\Pr[X_{|x|} = x])$, 而 $M_{S,X}$ 是一个算法, (在给定 x' 和 x 时) 先验证 x' 是否为 x 的正确编码, 再应用问题 S 的标准验证 (即 M_S)。我们将证明这种归约满足上述 3 个条件 (即高效性、正确性和支配性)。因此, 归约过程不是强制性地让目标分布 U' 具有原始分布 X 的结构, 而是将 X 的结构包含于归约后的实例中。要做到这一点, X 必须是简单的 (由定义 10.15)。

有了上述动机, 现在进行实际的证明, 即证明任意 $(S, X) \in \text{distNP}$ 可归约为 (S_u, U') 。以下的技术性引理是归约的基础。在该引理及后续结论中, 考虑概率空间 X 的 (累积) 分布函数是有利的。即考虑 $\mu(x) \stackrel{\text{def}}{=} \Pr[X_{|x|} \leq x]$, 且注意 $\mu: \{0,1\}^* \rightarrow [0,1]$ 是多项式时间可计算的 (因为 X 满足定义 10.15)。

编码引理^①

令 $\mu: \{0,1\}^* \rightarrow [0,1]$ 为多项式时间可计算的函数, 对任意 n , μ 在 $\{0,1\}^n$ 上单调非降 (即对任意 $x' < x'' \in \{0,1\}^n$, 有 $\mu(x') \leq \mu(x'')$)。对 $x \in \{0,1\}^n \setminus \{0^n\}$, 令 $x-1$ 表示在 n 比特串的词典序中位于 x 之前的串, 则存在一个编码函数 C_μ , 满足以下 3 个条件。

(1) 压缩。对任意 x , 有 $|C_\mu(x)| \leq 1 + \min\{|x|, \log_2(1/\mu'(x))\}$, 其中当 $x \notin \{0\}^*$ 时, $\mu'(x) \stackrel{\text{def}}{=} \mu(x) - \mu(x-1)$, 否则, $\mu'(0^n) \stackrel{\text{def}}{=} \mu(0^n)$ 。

(2) 高效编码。函数 C_μ 可在多项式时间内计算。

① 引理中的 $\{0,1\}^n$ 实际上针对任意给定的 n 值, 但是允许 n 变化 (并利用算法的标准渐近分析) 时高效性条件将更容易表述。

实际上, 如果使用 $\{0,1\}^*$ (而非 $\{0,1\}^n$) 上多项式时间计算的单调非增函数, 引理可能会更易于表述和证明。见习题 10.19 中的进一步讨论。

(3) 唯一解码。对任意 $n \in \mathbb{N}$, 当局限于 $\{0,1\}^n$ 时, 函数 C_μ 为双射 (即如果 $C_\mu(x) = C_\mu(x')$ 且 $|x| = |x'|$, 则 $x = x'$)

证明: 函数 C_μ 的定义如下。如果 $\mu'(x) \leq 2^{-|x|}$, 则 $C_\mu(x) = 0x$ (即此时将 x 作为对其自身的编码), 否则 (即 $\mu'(x) > 2^{-|x|}$ 时), $C_\mu(x) = 1z$, 其中 $|z| \leq \log_2(1/\mu'(x))$, 且 n 比特串到其编码的映射为双射。不严格地, 在选择 z 时, 要令其等于区间 $(\mu(x) - \mu'(x), \mu(x)]$ 中整数的最短的二进制表示。注意: 该区间的长度为 $\mu'(x)$, 而不同的区间是不相交的, 从而得到了期望的编码方式。细节如下。

主要考虑 $\mu'(x) > 2^{-|x|}$ 的情形, 并详述 z 的选择方法 (在编码 $C_\mu(x) = 1z$ 中)。如果 $x > 0^{|x|}$, 且 $\mu(x) < 1$, 则令 z 为 $\mu(x-1)$ 与 $\mu(x)$ 的二进制表示中的最长公共前缀。例如, 如果 $\mu(1010) = 0.10010$, $\mu(1011) = 0.10101111$, 则 $C_\mu(1011) = 1z$, 其中 $z = 10$ 。因此, 此时, $0.z1$ 在区间 $(\mu(x-1), \mu(x)]$ 内 (即 $\mu(x-1) < 0.z1 < \mu(x)$)。对于 $x = 0^{|x|}$, 令 z 为 0 与 $\mu(x)$ 的二进制表示的最长公共前缀, 此时, $0.z1$ 也在相应区间内 (即 $(0, \mu(x)]$)。最后, 对于使 $\mu(x) = 1$ 且 $\mu(x-1) < 1$ 的 x , 令 z 为 $\mu(x-1)$ 和 $1 - 2^{-|x|-1}$ 的二进制表示的最长公共前缀, 此时, $0.z1$ 仍在区间 $(\mu(x-1), \mu(x)]$ 内 (因为 $\mu'(x) > 2^{-|x|}$ 且 $\mu(x-1) < \mu(x) = 1$ 蕴含了 $\mu(x-1) < 1 - 2^{-|x|} < \mu(x)$)。注意: $\mu(x) = \mu(x-1) = 1$, 从而 $\mu'(x) = 0 < 2^{-|x|}$ 。

现在验证上述的 C_μ 满足引理中条件。从压缩条件开始。显然, 如果 $\mu'(x) \leq 2^{-|x|}$, 则 $|C_\mu(x)| = 1 + |x| \leq 1 + \log_2(1/\mu'(x))$ 。另一方面, 假设 $\mu'(x) > 2^{-|x|}$, 则主要考虑 $x > 0^{|x|}$ 且 $\mu(x) < 1$ 的情况。令 $z = z_1 z_2 \cdots z_l$ 为 $\mu(x-1)$ 与 $\mu(x)$ 的二进制表示的最长公共前缀, 则 $\mu(x-1) = 0.z0u$ 且 $\mu(x) = 0.z1v$, 其中 $u, v \in \{0,1\}^*$ 。我们推断

$$\mu'(x) = \mu(x) - \mu(x-1) \leq \left(\sum_{i=1}^l 2^{-i} z_i + \sum_{i=l+1}^{\text{poly}(|x|)} 2^{-i} \right) - \sum_{i=1}^l 2^{-i} z_i < 2^{-|z|}$$

且 $|z| < \log_2(1/\mu'(x)) \leq |x|$ 。因此, 在两种情况下均有 $|C_\mu(x)| \leq 1 + \min(|x|, \log_2(1/\mu'(x)))$ 。显然, 通过计算 $\mu(x-1)$ 和 $\mu(x)$, 可在多项式时间内求出 C_μ 。最后, 分别考虑上述两种情形 (即 $C_\mu(x) = 0x$ 和 $C_\mu(x) = 1z$), 可证明 C_μ 满足唯一解码条件。具体地, 在第二种情况下 (即 $C_\mu(x) = 1z$), 可以利用 $\mu(x-1) < 0.z1 \leq \mu(x)$ 的事实。□

为了得到一一映射的编码方式以便应用于不同长度的串, 对 C_μ 作如下增强, 即考虑 $C'_\mu(x) \stackrel{\text{def}}{=} (|x|, C_\mu(x))$, 实现时, 可令 $C'_\mu(x) = \sigma_1 \sigma_1 \cdots \sigma_l \sigma_l C_\mu(x)$, 其中 $\sigma_1 \sigma_2 \cdots \sigma_l$ 是 $|x|$ 的二进制表示。注意到, $|C'_\mu(x)| = O(\log|x|) + |C_\mu(x)|$, 且 C'_μ 是 $(\{0,1\}^* \text{ 上的 })$ 一一映射。

与 (S, X) 相关联的机器

令 μ 为与概率空间 X 相关的累积概率函数, M_S 为验证 S 中成员问题的多项式时间算法, M_S 使用充分的 NP-证据 (即 $x \in S$ 当且仅当存在 $y \in \{0,1\}^{\text{poly}(|x|)}$ 使得 $M(x, y) = 1$)。利用编码函数 C'_μ 引入算法 $M_{S, \mu}$, 目的是将分布问题 (S, X) 归约为 (S_μ, U') , 使得 $(S$ 中) 所有实例被映

射为一个三元组, 其中第一个元素为 $M_{S,\mu}$ 。 $M_{S,\mu}$ 的输入为 S 中一个实例 (在 C'_μ 下) 的编码以及相应实例属于 S 的证明, 算法验证证明及编码。即对输入的 x' 及 $\langle x, y \rangle$, $M_{S,\mu}$ 首先验证 $x' = C'_\mu(x)$, 再运行 $M_S(x, y)$ 来验证 $x \in S$ 。因此, 通过利用形如 $\langle x, y \rangle$ 且满足 $M_S(x, y) = 1$ 的证明, $M_{S,\mu}$ 可以验证集合 $S' = \{C'_\mu(x) : x \in S\}$ 的成员。(对实例 $C'_\mu(x)$)。^①

归约过程

将 (S 的) 实例 x 映射为三元组 $(M_{S,\mu}, C'_\mu(x), 1^{p(|x|)})$, 其中 $p(n) \stackrel{\text{def}}{=} p_S(n) + p_C(n)$, p_S 为表示 M_S 运行时间的多项式, p_C 为表示编码算法运行时间的多项式。

对归约的分析

我们要证明上述映射构成了 (S, X) 到 (S_μ, U') 的归约, 为此, 要验证符合定义 10.16 中的 3 个条件。

(1) 利用 C'_μ 在多项式时间计算这一事实 (且注意到 p 为多项式), 可证明上述映射可在多项式时间内计算。

(2) 前面指出, 对于输入的 $(x', \langle x, y \rangle)$, 机器 $M_{S,\mu}$ 接受该输入当且仅当 $x' = C'_\mu(x)$ 且 M_S 在 $p_S(|x|)$ 步内接受 (x, y) 。由于 $C'_\mu(x)$ 能唯一确定 x , 从而 $x \in S$ 当且仅当 $C'_\mu(x) \in S'$, 后者成立当且仅当存在串 y 使得 $M_{S,\mu}$ 在至多 $p(|x|)$ 步内接受 $(C'_\mu(x), \langle x, y \rangle)$, 从而, $x \in S$ 当且仅当 $(M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}) \in S_\mu$, 满足高效性条件。

(3) 为了验证支配性条件, 首先注意到上述映射为双射 (由于变换 $x \rightarrow C'_\mu(x)$ 是双射)。下一步, 注意到, 只需考虑在上述映射下有原像的 S_μ 的实例即可 (因为没有原像的实例显然满足支配性条件)。每个这种实例 (即映射的像) 是一个三元组, 其中第一个元素为 $M_{S,\mu}$, 第二个元素为 C'_μ 下的编码。由 U' 的定义, 对每个这种像 $\langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)} \rangle \in \{0, 1\}^n$, 有

$$\Pr[U'_n = \langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)} \rangle] = \binom{n}{2}^{-1} \cdot 2^{-(|M_{S,\mu}| + |C'_\mu(x)|)} > c \cdot n^{-2} \cdot 2^{-(|C_\mu(x)| + O(\log|x|))}$$

式中: $c = 2^{-|M_{S,\mu}| - 1}$ 为只依赖于 S 和 μ 的常量 (即对分布问题 (S, X))。因此, 对某个正多项式 q , 有

$$\Pr[U'_n = \langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)} \rangle] > 2^{-|C_\mu(x)|} / q(n) \quad (10.4)$$

根据 (编码引理中的) 压缩条件, 有 $2^{-|C_\mu(x)|} \geq 2^{-1 - \min(|x|, \log_2(1/\mu'(x)))}$, 从而

$$2^{-|C_\mu(x)|} \geq \Pr[X_{|x|} = x] / 2 \quad (10.5)$$

由于 x 是 $\langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)} \rangle$ 的唯一原像, 结合式 (10.4) 与式 (10.5), 可以证明支配性条件。

定理得证。 ■

^① 注意: $|y| = \text{poly}(|x|)$, 但是 $|x| = \text{poly}(|C'_\mu(x)|)$ 并不一定成立 (从而 S' 不一定属于 NP)。不过, 我们将看到, 后者是否成立并不重要。

思考

定理 10.17 的证明强调了一个事实, 即定理 2.19 的证明中使用的归约没有向被归约实例集引入更多的结构 (即没有将原始问题归约为目标问题的一个“具有高结构性的特例”)。换言之, 与更高级的最坏情况归约不同, 这种归约没有将“随机的”(即均匀分布的)实例映射为具有高级结构的实例 (其在均匀分布中出现的概率可忽略)。因此, 定理 2.19 证明中使用的归约足以将 distNP 中的任何分布问题归约为包含 S_u 及某个简单概率空间的分布问题 (见习题 10.20)。^①

然而, 定理 10.17 具有比上述结论更丰富的含义。即 distNP 中的任意分布问题可以归约为 S_u 中同一问题的分布版本。事实上, 在定理 10.17 的证明过程中, 最大的困难来自于要将任意简单概率空间 (可能并非均匀空间) 中的实例映射为其分布被一个简单概率空间支配制的实例 (即平均均匀空间 U')。

一旦证明了存在 distNP -完全问题, 就可以证明 (distNP 中) 其他问题的 distNP -完全性, 方法是将某个 distNP -完全问题归约为该问题 (并利用归约的传递性 (见习题 10.17))。从而, 定理 10.17 证明中的困难不再是实质性的。不幸的是, 产生了一个似乎更严重的困难: NP -完全性理论中几乎所有的已知归约都会向目标实例集引入更多的结构 (即它们实际上将问题实例归约为更具结构性的特殊情形)。进一步地, 产生的结构过于复杂, 以至于目标实例的分布不再是简单的 (在定义 10.15 的意义上)。以下将阐明, 目标实例的结构并不重要, 重要的是这种结构的复杂性。特别地, 如果上述归约是“单调”且“长度不变”的, 则目标实例的分布将足够简单 (即在定义 10.15 的意义上是简单的)。

命题 10.18 (distNP -完全性的充分条件) 假设 f 为集合 S 到集合 T 的 Karp-归约, 满足对任意 $x', x'' \in \{0, 1\}^*$, 以下两个条件成立:

(1) (f 是单调的): 如果 $x' < x''$, 则 $f(x') < f(x'')$, 其中不等式针对着字符串的标准词典序。^②

(2) (f 为长度不变的): $|x'| = |x''|$ 当且仅当 $|f(x')| = |f(x'')|$ 。

如果存在一个概率空间 X , 使得 (S, X) 为 distNP -完全问题, 则存在概率空间 Y , 使得 (T, Y) 也是 distNP -完全问题。

证明概要: 注意: f 的单调性蕴含着 f 为一一映射, 且对任意 x , 有 $f(x) \geq x$ 。另外, 以下将证明, f 可在多项式时间内求逆。直观上, f 既是单调的又可在多项式时间内计算蕴含着可由二分查找法求出一个原像。具体而言, 给定 $y = f(x)$, 可以将解区间不断减半来找到 x , 解区间被初始化为 $[0, y]$ (由于 $x \leq f(x)$)。注意: 如果 y 不是 f 的原像, 则查找结束时将检测到这一事实。

根据 f 为一一映射 (且长度不变) 的事实, 可以定义概率空间 $Y = \{Y_n\}_{n \in \mathbb{N}}$, 使得对任意 x , 有 $\Pr[Y_{|f(x)|} = f(x)] = \Pr[X_{|x|} = x]$ 。具体地, 令 $l(m) = |f(1^m)|$ 并注意到 l 是双射且单调非降, 定义

① 注意到, 这对大部分已知的 Karp-归约而言并不成立, 因为这些归约确实将随机实例映射为具有高级结构的实例。进一步地, 对于由习题 2.31 得到的归约, 也有同样的情形 (结构创设性)。

② 特别地, 如果 $|z'| < |z''|$, 则 $z' < z''$ 。回顾对于 $|z'| = |z''|$, 有 $z' < z''$ 当且仅当存在 $w, u', u'' \in \{0, 1\}^*$, 满足 $z' = w0u'$ 且 $z'' = w1u''$ 。

$$\Pr[Y_{|y|} = y] = \begin{cases} \Pr[X_{|x|} = x], & x = f^{-1}(y) \\ 0, & \text{存在 } m, \text{ 使得 } y \in \{0,1\}^{l(m)} \setminus \{f(x) : x \in \{0,1\}^m\} \\ 2^{-|y|}, & \text{其他 (即 } |y| \notin \{l(m) : m \in \mathbb{N}\} \text{)} \quad \textcircled{2} \end{cases}$$

显然, (S, X) 可归约为 (T, Y) (利用 Karp-归约 f , 根据 Y 的构造, 该归约也满足支配性条件)。因此, 利用 distNP 可归约为 (S, X) 的假设, 以及归约的传递性 (见习题 10.17), 可得到结论, distNP 中所有问题可归约为 (T, Y) 。要格外注意的是, 以下将证明, Y 是一个简单概率空间, 从而 (T, Y) 属于 distNP 。

不严格地说, Y 的简单性可结合 X 的简单性以及 f 的性质 (即 f 为单调、长度不变且多项式时间可求逆) 而得到。 f 的单调性及长度不变的特征蕴含了 $\Pr[Y_{|f(x)|} \leq f(x)] = \Pr[X_{|x|} \leq x]$ 。更一般地, 对任意 $y \in \{0,1\}^{l(m)}$, 有 $\Pr[Y_{l(m)} \leq y] = \Pr[X_m \leq x]$, 其中 x 为词典序中满足 $f(x) \leq y$ 的最大串 (并且事实上, 如果 $|x| < m$, 则 $\Pr[Y_{l(m)} \leq y] = \Pr[X_m \leq x] = 0$)。^①注意到, x 可在多项式时间内通过证明第一段中的求逆算法找到。因此, 可以找到足够多的 x 并计算 $\Pr[X_{|x|} \leq x]$, 从而求得 $\Pr[Y_{|y|} \leq y]$ 。利用 X 为简单概率空间的假设, 可得到 Y 也是简单空间 (命题得证)。□

充分 Karp-归约的存在性

命题 10.18 中暗示: 利用 (NP-完全) 集合 T 分布版本的 distNP -完全性的充分条件, 可以证明存在集合 S_u 到集合 T 的充分 Karp-归约, 即这种 Karp-归约是单调且长度不变的。其中长度不变的条件似乎易证明 (利用充分的填充), 而单调性条件显得更难。幸运的是, 单调性条件也可利用充分的填充 (或充分的“标记”, 见习题 2.30 和习题 10.21) 来证明。我们强调一个事实, 即存在充分填充 (或“标记”) 是集合 T 的固有性质。习题 10.21 将介绍一种方法, 将到“单调标记”集合 T 的任意 Karp-归约修改为 (到 T 的) 单调且长度不变的 Karp-归约。在习题 10.23 中, 为命题“所有自然的 NP-完全集合均为单调标记集”提供证据。结合上述所有这些事实, 可以得出结论, 任意自然的 NP-完全判定问题与简单概率空间相结合, 得到的分布问题是 distNP -完全的。为了详述这个命题, 我们用 Karp 在关于 NP-完全性的论文^[138]中提出的 21 个 NP-完全问题的相关 (正规的) 结论来陈述。

定理 10.19 (通用命题的温和版本) 参考文献文献[138]提出的 21 个 NP-完全问题中, 每一个都存在简单的概率空间, 使得与之结合后形成的分布问题是 distNP -完全的。

这里所说的问题包括 SAT、团及 3-着色问题。

10.2.1.3 概率版本

10.2.1.1 节中的定义可以扩展到随机计算中。例如, 对定义 10.14 进行扩展, 得到

定义 10.20 (tpcBPP 类) 对于概率算法 A , 布尔函数 f 以及时间受限的函数 $t: \mathbb{N} \rightarrow \mathbb{N}$, 如果存在输入 x , 或者 $A(x) \neq f(x)$, 或者 A 的运算超过了 $t(|x|)$ 步, 这两个事件之一成立的概率超过了 $1/3$, 则称串 x 是 A 关于 f 的 t -不良输入。如果存在多项式 p , 使得 X_n 为 A 关于 S

① 还要注意一种情况, 即 $|y|$ 不属于 l 的像集是很容易检测到的, 从而可以做相应的处理。

② 令 Y_n 为均匀分布是一种相当随意的选择, 其目标仅仅是为了保证 n -比特串上的“简单”概率分布。

特征函数的 p -不良输入的概率可以忽略, 则称 A 在概率多项式时间内典型地解 $(S, \{X_n\}_{n \in \mathbb{N}})$ 。将可在概率多项式时间内典型求解的分布问题类记作 tpcBPP 类。

可以对归约的定义做类似扩展。即在定义 10.16 中, 令 $M^T(x)$ 和 $Q(x)$ (分别在第 (2) 项、第 (3) 项中提到) 为随机变量而非固定对象。进一步地, 要求正确性 (对所有输入) 只以 $2/3$ 的概率成立, 这里概率空间只针对归约的内部随机数。随机归约在合成下满足封闭性, 且保持典型可行性 (见习题 10.24)。

随机归约中可将 distNP -完全问题用 (完全) 均匀概率空间表示。回顾定理 10.17 中证明了 (S_u, U') 的 distNP -完全性, 其中 U' 为半均匀的概率空间 (即 $\Pr[U'_n = \langle M, x, l' \rangle] = 2^{-(|M|+|x|)} / \binom{n}{2}$, 其中 $n = |\langle M, x, l' \rangle|$)。首先要注意, (S_u, U') 可以随机归约为 (S'_u, U'') , 其中 $S'_u = \{ \langle M, x, z \rangle : \langle M, x, l' \rangle \in S_u \}$ 且对任意 $\langle M, x, z \rangle \in \{0, 1\}^n$, 有 $\Pr[U''_n = \langle M, x, z \rangle] = 2^{-(|M|+|x|+|z|)} / \binom{n}{2}$ 。随机归约中包含 $\langle M, x, l' \rangle$ 到 $\langle M, x, z \rangle$ 的映射, 其中 z 在 $\{0, 1\}^l$ 中均匀选择。回顾 $U = \{U_n\}_{n \in \mathbb{N}}$ 表示均匀概率空间 (即 U_n 在长为 n 的串上均匀分布), 利用一种适当的编码, 可以得到以下命题。

命题 10.21 存在 $S \in \text{NP}$, 使得任意 $(S', X') \in \text{distNP}$ 可以随机地归约为 (S, U) 。

证明概要: 由上述讨论, 任意 $(S', X') \in \text{distNP}$ 可以随机归约为 (S'_u, U'') , 其中归约经过了 (S_u, U') 。因此, 只需将 (S'_u, U'') 归约为 (S''_u, U) 即可, 其中 $S''_u \in \text{NP}$ 的定义如下。串 $\text{bin}_l(|u|) \cdot \text{bin}_l(|v|) \cdot u \cdot v \cdot w$ 属于 S''_u 当且仅当 $\langle u, v, w \rangle \in S'_u$ 且 $l = \lceil \log_2 |uvw| \rceil + 1$, 其中 $\text{bin}_l(i)$ 表示整数 $i \in [2^{l-1}]$ 的 l 比特二进制编码 (即向码字中填充 0 使其总长度达到 l)。归约将 $\langle M, x, z \rangle$ 映射为串 $\text{bin}_l(|x|) \cdot \text{bin}_l(|M|) \cdot M \cdot x \cdot z$, 其中 $l = \lceil \log_2 (|M| + |x| + |z|) \rceil + 1$ 。注意: 这个归约满足定义 10.16 中的所有条件, 从而命题得证。□

10.2.2 研究分支

10.2.1 节的最大问题是只讨论了简单概率空间, 这就将“NP 的分布式版本”限制到 distNP 类 (定义 10.15)。10.2.1.1 节表明, 这种限制引出两个相对立的问题 (即 distNP 或者太宽或者太窄)。^①本节讨论对简单概率空间做出过多限制的情形, 从而使 $\text{distNP} \not\subseteq \text{tpcBPP}$ 的猜想过于强大 (这将意味着 distNP -完全性是典型情况困难性的一种较弱证据)。10.2.2.2 节提出对简单概率空间类一种有意义的扩展, 由此得到对 distNP 的相应扩展, 并且证明了, 如果对 distNP 的这种扩展不包含于 tpcBPP 中, 则 distNP 自身也不包含于 tpcBPP 中。结果, distNP -完全问题具有两种优势, 既属于更加受限的类 (即 distNP), 又保持了困难性 (只要扩展类中某个问题是困难的)。

10.2.2.1 节还提出了另一种扩展, 讨论范围由判定问题扩展到搜索问题。后者在实际应用中更加重要 (见 2.1.1 节), 且由实际应用提出的理论必须能处理搜索问题。

^① 一方面, 如果 distNP 的定义过于自由, 则 distNP 的成员含义可能并不如期望中那样丰富。另一方面, 如果对 distNP 作了过多限制, 则其中包含困难问题的猜想将变得很可疑。

预备知识

10.2.2.1 节中的技术细节需要读者熟悉唯一解的概念, 相关结论可参考 6.2.3 节。阅读 10.2.2.2 节则需熟悉附录 D.2 中的 Hash 函数。另外, 10.2.2.1 节实际上是 10.2.2.2 节的基础。

10.2.2.1 搜索与判定

与最坏情况复杂性相同, 搜索问题至少与判定问题具有同等的重要性。因此, 确实有必要在平均情况下讨论搜索问题。我们首先介绍 $\mathcal{PF}$ 和 $\mathcal{PC}$ (见 2.1.1 节) 的分布版本, 再讨论 $\text{tpc}\mathcal{P}$ 与 $\text{dist}\mathcal{NP}$ 的基本原理。

定义 10.22 ($\text{tpc}\mathcal{PF}$ 与 $\text{dist}\mathcal{PC}$ 类) 与 2.1.1 节类似, 只考虑具有多项式界限的搜索问题, 即二元关系 $R \subseteq \{0,1\}^* \times \{0,1\}^*$, 使得对某个多项式 q , 有 $(x,y) \in R$ 蕴含着 $|y| \leq q(|x|)$ 。回顾前面定义 $R(x) \stackrel{\text{def}}{=} \{y : (x,y) \in R\}$ 及 $S_R \stackrel{\text{def}}{=} \{x : R(x) \neq \emptyset\}$ 。

- 一个分布式搜索问题由多项式界限的搜索问题和一个概率空间构成。

- $\text{tpc}\mathcal{PF}$ 类中包含了所有可在多项式时间内典型求解的分布式搜索问题, 即 $(R, \{X_n\}_{n \in \mathbb{N}}) \in \text{tpc}\mathcal{PF}$, 如果存在算法 A 及多项式 p , 使得 A 对输入 x 的计算出错或运行多于 $p(n)$ 步的概率可以忽略, 其中 A 对 $x \in S_R$ 的计算出错是指 $A(x) \notin R(x)$, 而对 $x \notin S_R$ 出错是指 $A(x) \neq \perp$ 。

- 分布式搜索问题 (R, X) 属于 $\text{dist}\mathcal{PC}$, 如果 $R \in \mathcal{PC}$ 且 X 是简单的 (如定义 10.15)。

类似地, $\text{tpc}\mathcal{BPPF}$ 类中包含了所有可在概率多项式时间内典型求解 (见定义 10.20) 的分布式搜索问题。在判定问题中描述的分布式问题间的归约, 也可扩展到搜索问题。

幸运的是, 与最坏情况复杂性相同, 对分布式搜索问题的研究可“简化”为对分布式判定问题的研究。

定理 10.23 (搜索到判定的简化) $\text{dist}\mathcal{PC} \subseteq \text{tpc}\mathcal{BPPF}$ 当且仅当 $\text{dist}\mathcal{NP} \subseteq \text{tpc}\mathcal{BPP}$ 。进一步地, $\text{dist}\mathcal{NP}$ 中任意问题可归约为 $\text{dist}\mathcal{PC}$ 中某个问题, 且 $\text{dist}\mathcal{PC}$ 中任意问题也可随机地归约为 $\text{dist}\mathcal{NP}$ 中某个问题。

证明概要: 定理的后一部分类似于定理 2.6 的证明 (也可见定理 2.16 证明中的第一步)。事实上, 定理 2.6 证明中提出的 $\mathcal{NP}$ 到 $\mathcal{PC}$ 的归约可在这里进行扩展。具体地, 对任意 $S \in \mathcal{NP}$, 考虑关系 $R \in \mathcal{PC}$ 使得 $S = \{x : R(x) \neq \emptyset\}$, 并注意到对任意概率空间 X , 恒等变换将 (S, X) 归约为 (R, X) 。

在反方向, 即由 $\text{dist}\mathcal{NP} \subseteq \text{tpc}\mathcal{BPP}$ 证明 $\text{dist}\mathcal{PC} \subseteq \text{tpc}\mathcal{BPPF}$ 存在一个难点。回顾在定理 2.6 的证明中, 将搜索问题 $R \in \mathcal{PC}$ 归约为 $S'_R \stackrel{\text{def}}{=} \{\langle x, y' \rangle : \exists y'', s.t. (x, y'y'') \in R\} \in \mathcal{NP}$ 的成员判定。这里遇到的困难是对于输入 x , 该归约发出形如 $\langle x, y' \rangle$ 的询问, 其中 y' 为 $R(x)$ 中某个串的前缀。这些询问诱导出的分布可能不受任何简单分布支配。因此, 必须寻找别的归约。

作为热身, 先暂时假设 R (在定义 6.28 的意义上) 有唯一解, 即对任意 x , 有 $|R(x)| \leq 1$ 。此时, 易将搜索问题 $R \in \mathcal{PC}$ 归约为 $S''_R \in \mathcal{NP}$ 的成员判定, 其中 $\langle x, i, \sigma \rangle \in S''_R$ 当且仅当 $R(x)$ 中包含一个串, 其中第 i 比特等于 σ 。具体地, 对于输入 x , 归约发出询问 $\langle x, i, \sigma \rangle$, 其中 $i \in [l]$ ($l = \text{poly}(|x|)$) 且 $\sigma \in \{0,1\}$, 这也考虑到了确定集合 $R(x) \subseteq \{0,1\}^l$ 中的单个串 (无论是否 $|R(x)| = 1$)。关键在于这种归约可以将任意 (具有唯一解的) $(R, X) \in \text{dist}\mathcal{PC}$ 归约为

$(S_R'', X'') \in \text{dist}\mathcal{NP}$, 其中 X'' 将 x (在 X 下) 的概率等分到所有三元组 $\langle x, i, \sigma \rangle$, 即对任意 $i \in [l]$ 及 $\sigma \in \{0, 1\}$, 有 $\Pr[X''_{\langle x, i, \sigma \rangle} = \langle x, i, \sigma \rangle] = \Pr[X_{|x|} = x] / 2l$ 。

遗憾的是, 一般情况下, R 的解可能不唯一。然而, 利用定理 6.29 的证明思想, 可以克服这个困难。首先注意到, 即使 R 的解不唯一, 上述由 (具有唯一解的) 分布问题 $(R, X) \in \text{dist}\mathcal{PC}$ 的实例到 $(S_R'', X'') \in \text{dist}\mathcal{NP}$ 的实例间的映射仍满足高效性和支配性条件。(一般情况下) 正确性条件有可能不满足 (即当 $|R(x)| > 1$ 时, 可能无法恢复 $R(x)$ 中任意元素)。

回顾定理 6.29 证明的主要部分是一个随机归约, 将 R 的实例映射为形如 (x, m, h) 的三元组, 使得 m 在 $[l]$ 中均匀分布, h 在 Hash 函数类 H_l^m 中均匀分布, 其中 $l = \text{poly}(|x|)$ 而 H_l^m 如附录 D.2 所述。进一步地, 如果 $R(x) \neq \emptyset$, 则对所有的 $m \in [l]$ 以及 $h \in H_l^m$, 存在唯一的 $y \in R(x)$ 使得 $h(y) = 0^m$ 的概率为 $\Omega(1/l)$ 。定义 $R'(x, m, h) \stackrel{\text{def}}{=} \{y \in R(x) : h(y) = 0^m\}$, 可以得到由搜索问题 R 到搜索问题 R' 的随机归约, 且该归约以不可忽略的概率^①将有解的实例映射为有唯一解的实例。另外, 这个归约可将任意 $(R, X) \in \text{dist}\mathcal{PC}$ 归约为 $(R', X') \in \text{dist}\mathcal{PC}$, 其中 X' 将 x (在 X 下) 的概率分配到所有三元组 (x, m, h) , 使得对任意 $m \in [l]$ 和 $h \in H_l^m$, 有 $\Pr[X'_{\langle x, m, h \rangle} = \langle x, m, h \rangle] = \Pr[X_{|x|} = x] / (l \cdot |H_l^m|)$ (注意: 在适当的编码方式下, X' 确实是简单的)。

定理结合了上述两种归约。即首先应用 (R, X) 到 (R', X') 的随机归约, 再将得到的实例归约为相应判定问题 (S_R'', X'') 的实例, 其中 X'' 可通过修改 X' (而非 X) 而得到。最终合成的归约满足高效性和支配性条件, 且其正确的概率不可忽略。利用直接放大法 (见习题 10.24), 能将错误概率降低到可以忽略。□

10.2.2.2 简单分布与可采样分布

在定义简单概率空间时 (定义 10.15), 要求累积分布函数可在多项式时间内计算。称 $\mu: \{0, 1\}^* \rightarrow [0, 1]$ 为 $X = \{X_n\}_{n \in \mathbb{N}}$ 的累积分布函数, 如果对任意 $n \in \mathbb{N}$ 和 $x \in \{0, 1\}^n$, 有 $\mu(x) \stackrel{\text{def}}{=} \Pr[X_n \leq x]$, 其中不等号针对着 n -比特串的标准词典序。

10.2.1.1 节中表明, 要求累积分布函数可在多项式时间内计算严格限制了概率空间。进一步地, 这些简单空间在某种直观意义上似乎确实是“简单”的, 从而它们代表实际中可能出现的合理 (然而, 这一点有争议) 分布模型。鉴于担心对模型的这种限制仍过于严格 (从而使 $\text{dist}\mathcal{NP}$ -困难性成为典型情况困难性的较弱证据), 我们要寻找实际中可能出现的极大化分布模型。这就是多项式时间可采样的概率空间 (定义 10.24 中体现了)。极大化命题成立的基础是真实世界可以被建模为一种可行的随机过程 (而非任意过程)。这暗示了世界上所有对象都可视为用一种可行的随机过程采样而得到的。

定义 10.24 (可采样空间及 $\text{samp}\mathcal{NP}$ 类) 称概率空间 $X = \{X_n\}_{n \in \mathbb{N}}$ 是 (多项式时间) 可采样的, 如果存在一个概率多项式时间算法 A , 使得对任意 $x \in \{0, 1\}^*$, 有 $\Pr[A(1^{|x|}) = x] = \Pr$

① 称一个事件的概率是不可忽略的 (在相关参数下), 如果其取值大于某个正多项式的倒数。在随机归约中, 相关参数是指归约的输入长度。

$[X_{|x|=x}]$ 。用 $\text{samp}\mathcal{NP}$ 表示由 $\mathcal{NP}$ 中判定问题和可采样概率空间构成的分布问题类。

首先要注意所有简单概率空间都是可采样的(见习题 10.25), 因而, $\text{dist}\mathcal{NP} \subseteq \text{samp}\mathcal{NP}$ 。另一方面, 存在着并非简单的可采样概率空间(见习题 10.26)。

对分布问题的范围进行(由 $\text{dist}\mathcal{NP}$ 到 $\text{samp}\mathcal{NP}$ 的)扩展, 有助于阐述分布问题的完全性。首先注意, 易证明任意自然的 NP-完全问题在 $\text{samp}\mathcal{NP}$ 中存在一个 $\text{dist}\mathcal{NP}$ -困难的分布式版本(见习题 10.27)。另外, 还可证明所有自然的 NP-完全问题都有 $\text{samp}\mathcal{NP}$ -完全的分布式版本(在两种情况下, “自然”意味着相应的 Karp-归约不会对输入进行压缩, 与命题 10.18 相比, 这是一个较弱的条件)。

定理 10.25 (smpNP-完全性) 假设 $S \in \mathcal{NP}$ 且 $\mathcal{NP}$ 中任意集合可以利用一个无压缩的 Karp-归约归约为 S 。则存在一个多项式时间可采样空间 X , 使得 $\text{samp}\mathcal{NP}$ 中任意问题可以归约到 (S, X) 。

定理 10.25 的证明基于一个事实, 即存在一个多项式时间可采样空间, 能支配所有的多项式时间可采样空间。该空间的存在性依赖于通用(采样)机的概念。细节见习题 10.28。

定理 10.25 为 $\text{samp}\mathcal{NP}$ -完全性提供了丰富的理论, 但是并未将该理论与前面提出的 $\text{dist}\mathcal{NP}$ -完全性理论(图 10.1)相关联。以下定理做到了这一点, 其中断言 $\text{samp}\mathcal{NP}$ 中典型困难问题的存在性蕴含了其在 $\text{dist}\mathcal{NP}$ 中的存在性。

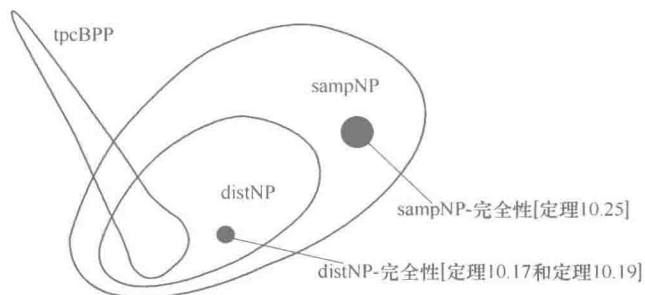


图 10.1 两种类型的平均情况完全性

定理 10.26 (smpNP-完全性与 $\text{dist}\mathcal{NP}$ -完全性) 如果 $\text{samp}\mathcal{NP}$ 不包含于 tpcBPP 中, 则 $\text{dist}\mathcal{NP}$ 也不包含于 tpcBPP 中。

因此, P-vs-NP 问题的两种“典型情况复杂性”版本是等价的。即如果 NP 的某个“可采样分布”版本不是典型可行的, 则 NP 的某个“简单分布”版本也不是典型可行的。特别地, 如果 $\text{samp}\mathcal{NP}$ -完全问题不包含于 tpcBPP 中, 则 $\text{dist}\mathcal{NP}$ -完全问题也不包含于 tpcBPP 中。

如果 $\text{samp}\mathcal{NP}$ 可以(随机地)归约为 $\text{dist}\mathcal{NP}$ (即如果 $\text{samp}\mathcal{NP}$ 中每个问题都可归约(在定义 10.16 的随机化版本下)为 $\text{dist}\mathcal{NP}$ 中的某个问题), 则可以直接证明上述断言成立。然而, 我们并不知道这种归约是否存在。不过, 定理 10.26 的证明中蕴含了分布问题间一种更宽松的归约。

证明概要: 证明如果 $\text{dist}\mathcal{NP}$ 包含在 tpcBPP 中, 则 $\text{samp}\mathcal{NP}$ 也包含于 tpcBPP 中。利用定理 10.23 及习题 10.29, 可以证明 distPC 包含于 tpcBPPF 中, 从而 distPC 的可采样版本, 记作 sampPC , 也包含于 tpcBPPF 中。证明思想是在随机归约的弱化版本下, sampPC 中任意问题可归约为 distPC 中的某个问题。不严格地说, (随机归约)这种弱化的概念只要求正确性和支配性(定义 10.16 (应用于随机归约中))在归约的概率空间中以不可忽略的概

率满足即可。^①在讨论分布式搜索问题时，首先给出上述归约的定义。

教学注释：以下的证明相当复杂，最好作为扩展阅读。其主要思想与定理 8.11 目前已知证明的核心思想有关。鉴于这种思想的大量应用，有必要理解以下证明过程。

定义：分布问题 (R, X) 到 (T, Y) 的弱化归约是一个概率多项式时间预言机 M ，定义簇 $\{\Omega_x \subseteq \{0,1\}^{m(|x|)}\}$ ，其中 $x \in \{0,1\}^*$ ， $m(|x|) = \text{poly}(|x|)$ 表示 M 对于输入 x 产生的内部随机数数量上限，则 M 关于 $\{\Omega_x\}$ 满足以下条件：

(Ω_x 的) 密度：存在不可忽略的函数 $\rho: \mathbb{N} \rightarrow [0,1]$ (即 $\rho(n) > 1/\text{poly}(n)$)，使得对任意 $x \in \{0,1\}^*$ ，有 $|\Omega_x| \geq \rho(|x|) \cdot 2^{m(|x|)}$ 。

正确性 (关于 Ω_x)：对任意 $r \in \Omega_x$ ，归约可得到一个正确结果，即如果 $R(x) \neq \emptyset$ ，则 $M^T(x, r) \in R(x)$ ，否则， $M^T(x, r) = \perp$ ，其中 $M^T(x, r)$ 表示 M 对于输入 x ，内部随机数 r ，以及预言 T 的运行结果。

支配性 (关于 Ω_x)：存在正多项式 p ，使得对任意 $y \in \{0,1\}^*$ 及任意 $n \in \mathbb{N}$ ，有

$$\Pr[Q'(X_n) \ni y] \leq p(|y|) \cdot \Pr[Y_{|y|} = y]$$

式中： $Q'(x)$ 为定义于 Ω_x 上的随机变量，表示 M 对于输入 x ， Ω_x 中随机数以及 T 的预言访问，发出的询问集合。即在定义 $Q'(x)$ 时，均匀选择 $r \in \Omega_x$ ，并考虑 M 对于输入 x ，内部随机数 r ，以及预言访问 T 所发出的询问集合 (另外，与定义 10.16 相同，还要求归约发出的询问不能太短)。

可以验证，这种弱化归约能保持典型可行性，即对 $R \in \mathcal{PC}$ ，如果存在 (R, X) 到 (T, Y) 的弱化归约，且 (T, Y) 属于 tpcBPPF ，则 (R, X) 也属于 tpcBPPF 。一个重要的观察是，在分析中可能不会考虑一种情况，即对输入 x ，归约中选择的随机数不在 Ω_x 中。事实上，此时发出的询问可能是非典型的，而收到的应答也可能是错误的，但这并不重要。重要的是，对于输入 x ，归约选择的随机数属于 Ω_x 的概率不可忽略，且生成“关于 Y 的典型”询问的概率也不可忽略 (根据放宽后的支配性条件)。典型地求解 (T, Y) 的算法可对这种典型询问做出正确应答，且如果 x 有解，则从这些应答中将得到 x 的正确解 (根据弱化的正确性条件)。因此，如果 x 有解，则归约输出正确解的概率不可忽略。另一方面，归约永远不会输出错误解 (即使所用的随机数不在 Ω_x 中)，因为不正确的解可以用 $R \in \mathcal{PC}$ 检测出来。

我们的目标是对任意 $(R, X) \in \text{sampPC}$ ，提供一个弱化的、从 (R, X) 到相关问题 (R', X') 的归约 (采用习惯写法 $X = \{X_n\}_{n \in \mathbb{N}}$ 及 $X' = \{X'_n\}_{n \in \mathbb{N}}$)。

一种过于简单的情形

对于初学者，假设 X_n 在某个集合 $S_n \subseteq \{0,1\}^n$ 上均匀分布，且存在多项式时间计算且可逆的映射 $\mu: S_n \rightarrow \{0,1\}^{l(n)}$ ，其中 $l(n) = \log_2 |S_n|$ 。于是，将 x 映射为 $1^{|x|-l(|x|)}0\mu(x)$ ，可以得到由 (R, X) 到 (R', X') 的归约，其中 X'_{n+1} 在 $\{1^{n-l(n)}0v: v \in \{0,1\}^{l(n)}\}$ 上均匀分布且

^① 注意：两个特定分布问题间这种弱化归约的存在性并不表明存在相应 (标准平均情况下的) 归约。特别是，虽然标准的正确性 (对 $\mathcal{PC}$ 中问题) 可以通过重复调用归约而得以保证，但这个过程并不能保证不违背标准支配性条件。

$R'(1^{n-l(n)}0v) = R(\mu^{-1}(v))$ (或等价地, $R(x) = R'(1^{|x|-l(|x|)}0\mu(x))$)。注意: X' 是一个简单空间, 且 $R' \in \mathcal{PC}$, 因此, $(R', X') \in \text{distPC}$ 。还要注意, 上述映射实际上是一个合法归约 (即满足高效性、正确性以及支配性条件)。因此, (R, X) 可归约为 distPC 中的问题 (且实际上在这里没有使用弱化)。

一种简单但更具指导意义的情形

下面不再假设存在一个多项式时间计算且可逆的映射 $\mu: S_n \rightarrow \{0,1\}^{l(n)}$, 但仍假设 X_n 在某个集合 $S_n \subseteq \{0,1\}^n$ 上均匀分布, 并假设 $|S_n| = 2^{l(n)}$ 易计算 (由 n 出发)。此时可利用一个随机选择的, 将 n 比特串映射为 $l(n)$ 比特串的 Hash 函数 h , 对 $x \in \{0,1\}^n$ 求其像。即随机地将 x 映射为 $(h, 1^{n-l(n)}0h(x))$, 其中 h 在适当的 Hash 函数簇 $H_n^{l(n)}$ (见附录 D.2) 中均匀选择。这就要求重新定义 R' , 使得 $R'(h, 1^{n-l(n)}0v)$ 对应于 S_n 中 v 在 h 下的原像。假设 h 是 S_n 到 $\{0,1\}^{l(n)}$ 的一一映射, 可以定义 $R'(h, 1^{n-l(n)}0v) = R(x)$, 使得 x 为唯一满足 $x \in S_n$ 且 $h(x) = v$ 的串, 其中条件 $x \in S_n$ 可通过生成 x 的采样程序使用的内部随机数来验证。将 X 的采样程序记作 S , 并令 $S(1^n, r)$ 表示 S 对于输入 1^n 及内部随机数 r 的输出, 则可以重新定义 R' 为

$$R'(h, 1^{n-l(n)}0v) = \left\{ \langle r, y \rangle : h(S(1^n, r)) = v \wedge y \in R(S(1^n, r)) \right\} \quad (10.7)$$

注意: 当 $S(1^{|x|}, r) = x$ 时, $\langle r, y \rangle \in R'(h, 1^{|x|-l(|x|)}0h(x))$ 是 $y \in R(x)$ 的一个解, 但是否则 “一切都有可能发生” (因为 y 将成为 $S(1^{|x|}, r) \neq x$ 的一个解)。虽然 h 不一定是 S_n 到 $\{0,1\}^{l(n)}$ 的一一映射, 但对任意 $x \in S_n$ 及所有可能的 h , $h(x)$ 在 S_n 中有唯一原像的概率为常数 (见定理 6.29 的证明)。此时, $\langle r, y \rangle \in R'(h, 1^{|x|-l(|x|)}0h(x))$ 蕴含着 $S(1^{|x|}, r) = x$ (这又蕴含着 $y \in R(x)$)。我们断言, 对于均匀选择的 $h \in H_{|x|}^{l(|x|)}$, 由 x 到 $(h, 1^{n-l(n)}0h(x))$ 的随机映射可得到 (R, X) 到 (R', X') 的弱化归约, 其中 X'_n 在 $H_n^{l(n)} \times \{1^{n-l(n)}0v : v \in \{0,1\}^{l(n)}\}$ 上均匀分布。当然, 这一断言针对的归约 (对于输入 x , 发出询问 $(h, 1^{n-l(n)}0h(x))$ 且) 当预言应答为 $\langle r, y \rangle$ 且 $y \in R(x)$ 时, 则返回 y 。

为了证明这个断言, 考虑集合 Ω_x , 对于所选择的 $h \in H_{|x|}^{l(|x|)}$, 其中 x 为 S_n 中 $h(x)$ 在 h 下的唯一原像 (即 $|\{x' \in S_n : h(x') = h(x)\}| = 1$)。此时 (即 $h \in \Omega_x$ 时), $\langle r, y \rangle \in R'(h, 1^{|x|-l(|x|)}0h(x))$ 蕴含着 $S(1^{|x|}, r) = x$ 且 $y \in R(x)$, 从而证明了 (弱化的) 正确性条件。证明 (弱化的) 支配性条件时, 注意到, $\Pr[X_n = x] \approx 2^{-l(|x|)}$, x 以概率 $1/|H_{|x|}^{l(|x|)}|$ 被映射为 $(h, 1^{|x|-l(|x|)}0h(x))$, 且 x 是 $(h, 1^{|x|-l(|x|)}0h(x))$ 在映射下的唯一原像 (在所有使得 $h \in \Omega_x$ 的 $x' \in S_n$ 中)。

在继续证明之前, 要强调将 X_n 映射为 $l(n)$ 比特串的 Hash 函数的重要性。一方面, 该映射是 “充分地” 一一映射, 从而 (以常数概率) 由实例 Hash 值 (即 $h(x)$) 的解可得到原始实例 (即 x) 的解。这就保证了归约的正确性。另一方面, 对于典型的 h , X_n 到 $h(X_n)$ 的映

射几乎是均匀地覆盖了相关值域。这保证了归约满足支配性条件。注意：这两种要求是矛盾的，而正确的 l 值则可以同时满足两种要求，即双射条件要求 $l(n) \geq \log_2 |S_n|$ ，而几乎均匀覆盖的条件要求 $l(n) \leq \log_2 |S_n|$ 。还要注意，对 X_n 的支集中任意 x ，有 $l(n) = \log_2 (1/\Pr[X_n = x])$ ，这是一般情况下的主要取值。

一般情况

最后，去掉 X_n 在 $\{0,1\}^n$ 的某个子集上均匀分布这一假设。我们知道的所有信息是存在一个概率多项式时间（“采样”）算法 S ，使得 $S(1^n)$ 与 X_n 具有相同分布。在这种（一般）情况下，将 (R, X) 的实例根据其概率求映射，使 x 被映射为 (R') 的一个实例，其中包含 $(h, h(x))$ 及其他信息，这里 h 为随机的 Hash 函数，将 n 比特串映射为 l_x 比特串，满足

$$l_x \stackrel{\text{def}}{=} \left\lceil \log_2 \left(1/\Pr[X_{|x|} = x] \right) \right\rceil \quad (10.8)$$

由于（在一般情况下） X_n 的支集中可能有多于 2^{l_x} 个串，因此需要对目标实例进行扩充，以保证它是与 x 唯一相关的实例。主要思想是在 x 到 $(h, h(x))$ 的映射中加入其他信息，从而将 X_n 限制为由至少以概率 2^{-l_x} 出现的串。实际上，对 X_n 进行这种限制之后， $h(X_n)$ 的值就唯一确定了 X_n 。

令 $q(n)$ 表示 S 的随机复杂度， $S(1^n, r)$ 表示 S 对输入 1^n 及内部随机数 $r \in \{0,1\}^{q(n)}$ 的输出，则可以随机地将 x 映射为 $(h, h(x), h', v')$ ，其中 $h: \{0,1\}^{|x|} \rightarrow \{0,1\}^{l_x}$ 和 $h': \{0,1\}^{q(|x|)} \rightarrow \{0,1\}^{q(|x|)-l_x}$ 为随机的 Hash 函数， $v' \in \{0,1\}^{q(|x|)-l_x}$ 服从均匀分布。重新定义的搜索问题 R' 的实例 (h, v, h', v') 的解为一个对 $\langle r, y \rangle$ ，使得 $h(S(1^n, r)) = v \wedge h'(r) = v'$ ，且 $y \in R(S(1^n, r))$ 。我们将看到，这种扩充保证了归约的目标实例 $(h, h(x), h', v')$ 的解以常数概率（对所有选择的 h, h', v' ）对应于原始实例 x 的解。

上述讨论中假设对于输入 x ，可以高效确定 l_x ，但这个假设无法证实。因此，作为代替，在 $\{0,1,\dots,q(|x|)\}$ 中均匀选择 l ，从而选择正确值的概率不可忽略（即 $\Pr[l = l_x] = 1/(q(|x|)+1) = 1/\text{poly}(|x|)$ ）。为了更清晰，令 n 和 l 在目标实例中是明确的。因此，将 $x \in \{0,1\}^n$ 随机地映射为 $(1^n, l', h, h(x), h', v') \in \{0,1\}^{n'}$ ，其中 $l \in \{0,1,\dots,q(n)\}$ ， $h \in H_n^l$ ， $h' \in H_{q(n)-l}^{q(n)-l}$ ，且 $v' \in \{0,1\}^{q(n)-l}$ 在相应集合中均匀分布。^①这个映射可将 (R, X) 归约为 (R', X') ，其中 R' 和 $X' = \{X'_{n',l}\}_{n' \in \mathbb{N}}$ 需（再次）重新定义。具体地，令

$$R'(1^n, l', h, v, h', v') = \left\{ \langle r, y \rangle : h(S(1^n, r)) = v \wedge h'(r) = v' \wedge y \in R(S(1^n, r)) \right\}$$

且 $X'_{n',l}$ 为每个 $X_{n,l}$ （ $l \in \{0,1,\dots,n\}$ ）分配相同的概率，其中每个 $X_{n,l}$ 同构于 $H_n^l \times \{0,1\}^l \times H_{q(n)-l}^{q(n)-l} \times \{0,1\}^{q(n)-l}$ 上的均匀分布。注意：实际上， $(R', X') \in \text{distPC}$ 。

① 在其他地方，利用适当的编码使归约将长度相同的串映射为长度相同的串（即 n 比特串被映射为 n' 比特串，其中 $n' = \text{poly}(n)$ ）。

例如，可将 $(1^n, l', h, h(x), h', v')$ 映射为 $1^n 01^l 01^{q(n)-l} 0 \langle h \rangle \langle h(x) \rangle \langle h' \rangle \langle v' \rangle$ ，其中每个 $\langle w \rangle$ 表示由长为 $(n' - (n + q(n) + 3))/4$ 的串对 w 的编码。

在分析上述随机映射时, 考虑正确选择的 l , 即对输入 x , 主要考虑 $l=l_x$ 的情形。此时 (正如以下将证明的), 对于所有 h, h' 和 v' , X_n 支集中实例 x 是唯一被映射为 $(1^n, l^x, h, h(x), h', v')$ 的值, 并满足 $\{r: h(S(1^n, r)) = h(x) \wedge h'(r) = v'\} \neq \emptyset$, 这一事件发生的概率为常数, 从而 (对这样的 h, h' 和 v'), 任意解 $\langle r, y \rangle \in R'(1^n, l^x, h, h(x), h', v')$ 满足 $S(1^n, r) = x$ 且 $y \in R(x)$ 。因此, 这意味着 (弱化的) 正确性条件满足。(弱化的) 支配性条件也满足, 因为 (对于这样的 h, h' 和 v' 以及 $l=l_x$), X_n 被映射为 $(1^n, l^x, h, h(x), h', v')$ 的概率近似等于 $\Pr[X'_{n, l_x} = (1^n, l^x, h, h(x), h', v')]$ 。

现在分析概率, 对所有的 h, h' 和 v' , 实例 x 是 X_n 的支集中唯一被映射为 $(1^n, l^x, h, h(x), h', v')$ 的值, 且满足 $\{r: h(S(1^n, r)) = h(x) \wedge h'(r) = v'\} \neq \emptyset$ 。首先, 注意到 $|\{r: S(1^n, r) = x\}| \geq 2^{q(n)-l_x}$, 因此, 对于 $h' \in H_{q(n)}^{q(n)-l_x}$ 和 $v' \in \{0, 1\}^{q(n)-l_x}$, 以常数概率存在 r 满足 $S(1^n, r) = x$ 及 $h'(r) = v'$ 。进一步地, 对于 $h' \in H_{q(n)}^{q(n)-l_x}$ 和 $v' \in \{0, 1\}^{q(n)-l_x}$, 以常数概率至多有 $O(2^{l_x})$ 个串 r 满足 $h'(r) = v'$ 。固定 h' 和 v' , 令 $S_{h', v'} = \{S(1^n, r): h'(r) = v'\}$, 且注意, 对所有 $h \in H_n^{l_x}$, x 是 $S_{h', v'}$ 中在 h 下唯一被映射为 $h(x)$ 的串的概率为常数。因此, 对所有的 h, h' 和 v' , 实例 x 是 X_n 支集中唯一被映射为 $(1^n, l^x, h, h(x), h', v')$ 的值, 并满足 $\{r: h(S(1^n, r)) = h(x) \wedge h'(r) = v'\} \neq \emptyset$, 这一事件发生的概率也为常数。定理得证。 $\square$

思考

定理 10.26 暗示了, 如果 sampNP 不包含于 tpcBPP 中, 则任一 distNP -完全问题也不属于 tpcBPP 。这意味着, 一些分布问题在可采样分布下的困难性蕴含了另一些分布问题在简单分布下的困难性。进一步地, 由命题 10.21 可知, 这蕴含了分布问题在均匀分布下的困难性。因此, 关于某些最大范围分布的困难性 (这包含实际中可能出现的所有分布) 蕴含了在一个简单分布下的困难性 (这是最简单的一种)。

与单向函数的关系

注意: 单向函数的存在性 (见 7.1 节) 蕴含了 sampPC 中存在一些不属于 tpcBPPF 的问题 (这又蕴含了 distPC 中也存在这样的问题)。具体而言, 对于一个保持长度的单向函数 f , 考虑分布式搜索问题 $(R_f, \{f(U_n)\}_{n \in \mathbb{N}})$, 其中 $R_f = \{(f(r), r): r \in \{0, 1\}^*\}$ 。^① 另一方面, $\text{sampPC} \setminus \text{tpcBPPF}$ 中问题的存在性是否蕴含着单向函数的存在性尚且未知。特别地, $\text{sampPC} \setminus \text{tpcBPPF}$ 中存在问题 (R, X) 这表明对搜索问题 R 生成困难实例是可行的, 而单向函数的存在性表明, 可以生成实例及其解构成的对, 且该实例为难解的 (见 7.1.1 节)。事实上, 这里的缝隙是指困难实例是否可以与其解一起高效地生成。由此将上述观点描绘为图 10.2, 其中较低层次表示似乎更弱的假设。

^① 注意: 在 f 为 $\{0, 1\}^n$ 上置换的特殊情况下, 分布 $f(U_n)$ 是均匀的。

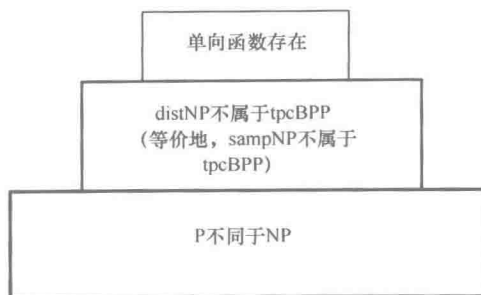


图 10.2 最坏情况与平均情况下的假设

本章注释

本章提出了对计算问题的两种不同弱化。第一种是近似，第二种是在平均情况下的分析。我们阐明了各种自然的近似概念都可以纳入标准框架中，其中承诺问题（见 2.4.1 节）框架是最不标准的一个（但它足以处理缝隙问题和属性检测）。相反，对平均情况复杂度的研究要求引入新的概念体系并处理各种定义问题。

这里自然产生一个问题，弱化要求有什么好处。这一问题在近似情况下有多种回答：一些例子放宽要求后会受益良多（即对困难问题进行了可行的弱化），而对其他一些例子而言，弱化毫无用处（即使是极端的弱化也与原始版本难度相当）。在平均情况复杂性中，否定的一面似乎占上风（至少在更系统化的意义上）。特别地，假设单向函数存在时，任意自然的 NP-完全问题都有一个（典型）困难的分布式版本，该版本涉及到一个可采样概率空间（并且事实上甚至是简单概率空间）。进一步地，此时，NP 中某些问题的分布版本在均匀分布下是困难的。

10.2.2.3 近似

以下的文献说明十分简洁，其中忽略了一些重要工作（包括本章提到的一些结论）。感兴趣的读者可以参考相关综述。

搜索或优化

在阐述了许多优化问题的 NP-完全性之后，对于近似算法的兴趣会增加。但也有一些例外（最著名的是参考文献[179]），对这类问题复杂性的系统研究曾一度停滞，直到 Feige, Goldwasser、Lovász 和 Safra^[73]发现了“PCP 关联”（见 9.3.3 节）。事实上，对于最大团、最大可满足，以及线性方程最大化问题的相对“紧凑”的不可近似性结论是由 Håstad^[116, 117]基于 PCP 及其与近似性之间联系的工作而给出的（如可见参考文献[16, 15, 29, 74, 185]）。具体地，定理 10.5 来自于参考文献[116]，^①而定理 10.8 和定理 10.9 来自于参考文献[117]。关于不可近似性，已知的最好结论是最小顶点覆盖（见定理 10.7），它来自于参考文献[69]，但是我们对它紧凑性有所怀疑（见参考文献[143]）。参考文献[179]中定义并给出了近似问题间的归约，见习题 10.7，其中介绍了参考文献[179]中引入的一种主要技术。对于近似算法及问题的一般性结论（如 10.1.1 节所讨论的），感兴趣的读者可以见参考文献[122]中收集的综述。参考文献[64]中给出了 NP 优化问题概述。

① 也可见于参考文献[244]。

另一种不同类型的近似问题通常与搜索问题相关,涉及到对解的个数求近似。6.2.2 节专门讨论了这类近似问题。注意可以利用 10.1.1 节中的定义框架来得到对近似计数问题更系统化的研究(如缝隙问题的概念、多项式时间近似方案等)。

属性检测

属性检测最早由 Rubinfeld 和 Sudan^[195]提出,后来又被 Goldreich、Goldwasser 和 Ron 等人^[97]重新发起。参考文献[195]的重点是代数性质,如低次多项式,而参考文献[97]的重点是图论性质(定理 10.12 来自参考文献[97])。参考文献[103]中引入了有界度数图论模型,定理 10.13 综合了参考文献[42, 103, 104]中的结论。这个领域的综述可以见参考文献[77, 194]。

平均情况复杂度

Levin 最早研究了平均情况复杂度^[154],特别地,他证明了定理 10.17。鉴于原始参考文献[154]十分简洁,建议感兴趣的读者阅读参考文献[89]中的综述,其中更详细地解释了 Levin 给出的定义,并讨论了概念背后的深刻含义(综述参考文献[89]还简要说明了该领域的研究进展)。

10.2.1.1 节中还指出,本章使用了原始定义的不同版本。特别地,我们对“典型情况可行性”的定义不同于最初的“平均情况可行性”概念,其中完全去掉了特殊实例,并甚至允许算法对这些实例失效(而不仅仅是运行超时)。研究者还提出了另外几种定义,作为参考文献[44]中一般情况的特例。

在 10.2.1.2 节中,我们指出,虽然 Levin 的原始论文^[154]证明了存在 distNP -完全问题(见定理 10.17),但是直到 20 多年后,参考文献[158]才证明了所有自然的 NP-完全问题都存在 distNP -完全版本(见定理 10.19)。

10.2.2 节的基础是参考文献[30, 127]。具体地,定理 10.23(或搜索到判定问题的归约)来自参考文献[30],另外,参考文献[30]中还引入了 sampNP 类。参考文献[127]证明了定理 10.26 的一个版本,我们的证明采用了其中的思想,这与定理 8.11(在[118]中证明)的证明思想十分接近。

我们知道,如果 sampNP 中包含困难问题,则 disNP 中也存在一些困难问题。然而,这些分布类问题似乎并非自然(它们或者涉及到普通的判定问题,如 S_u ,或涉及某个具有较强结构的概率空间(见定理 10.19))。将更自然的判定问题(如 SAT 或团)与更自然的概率空间相结合构造 distNP -完全问题,是一个公开问题。

习 题

10.1 (广义 TSP) 对任意函数 g ,证明以下近似问题是 NP-困难的。给定一个用距离矩阵描述的广义 TSP 实例 I ,求一条旅行路线,长度至多为最短路线长度的 $g(I)$ 倍。具体地,证明当 $g(I) = \exp(|I|^{0.99})$,且实例中所有距离均为正整数时,结论成立。

提示:利用从 Hamiltonian 回路到 TSP 的归约。具体地,将实例 $G = ([n], E)$ 归约为一个 $n \times n$ 矩阵 $D = (d_{i,j})_{i,j \in [n]}$,使得当 $\{i, j\} \in E$ 且 $d_{i,j} = 1$ 时, $d_{i,j} = \exp(\text{poly}(n))$ 。

10.2 (三角不等式下的 TSP) 对城市间距离满足三角不等式的一类特殊 TSP 问题, 构造一个多项式时间的 2-因子近似算法。

提示: 首先要注意, 任意旅行路线的长度下限是相应加权图中最小生成树的重量。其次要注意, 从这样一棵树出发, 可以得到多次经过某些点的一条路线(长度为该树重量的 2 倍)。三角不等式保证了通过“抄近路”而避免多次经过一个点时, 旅行路线的长度不会增加。

10.3 (定理 10.5 的弱化版本) 利用定理 9.16 证明, 对于常数 $0 < a < b < 1$, 在令 $L(N) = N^b$ 以及 $s(N) = N^a$ 时, $\text{gapClique}_{L,s}$ 是 NP-困难的。

提示: 从定理 9.16 出发, 利用扩张随机漫游发生器(命题 8.29) 得到一个 PCP 系统, 具有对数的随机性复杂度和询问复杂度, 以至多 $1/n$ 的概率接受长为 n 的“否”实例。利用 FGLSS-归约(习题 9.18), 并注意到 x 被映射到一个规模为 $\text{poly}(|x|)$ 的图, 使得“是”与“否”实例间的缝隙至少是 $|x|$ 的因子。

10.4 (定理 10.7 的弱化版本) 利用定理 9.16 证明, 对某个常数 $0 < s < L < 1$, 问题 $\text{gapLin}_{s,L}$ 是 NP-困难的。

提示: 注意: 将定理 9.16 与习题 9.18 相结合, 蕴含了对某个常数 $b < 1$, $\text{gapClique}_{L,s}$ 是 NP-困难的, 其中 $L(N) = b \cdot N$, 而 $s(N) = (b/2) \cdot N$ 。利用团、独立集与顶点覆盖问题间的关系, 断言得证。

10.5 (定理 10.9 的弱化版本) 利用定理 9.16 证明, 对某个常数 $0.5 < s < L < 1$, $\text{gapClique}_{L,s}$ 是 NP-困难的。

提示: 回顾定理 9.16 和定理 9.21, 缝隙问题 $\text{gapSAT}_\varepsilon^3$ 是 NP-困难的。注意: 即使我们将实例限定为每个子句恰含 3 个变量(不一定不同)时, 结论仍成立。应用习题 2.24 中的归约, 并注意到, 对任意赋值 τ , 由 τ 满足的子句被映射为 7 个等式, 其中恰有 3 个是 τ 不满足的, 而 τ 不满足的子句被映射为 7 个等式, 在赋值 τ 下均不满足。

10.6 (无 PCP 时的自然不可近似性) 与 10.1.1.2 节中的不可近似性结论相反, 证明如下缝隙问题的 NP-完全性时, 可以不使用 PCP 系统。问题实例为 $\text{GF}(2)$ 上的二次方程组(如习题 2.25), “是”实例为有解的方程组, “否”实例为任意赋值都使至少 $1/3$ 的方程不满足的方程组。

提示: 如习题 2.25 所述, 当给定这样一个二次方程组时, 判定是否存在一种满足所有方程的赋值是 NP-困难的。利用一个充分的小偏移量发生器(见 8.5.2 节), 构造上述问题到自身的放大归约(见 9.3.3 节)。具体地, 如果输入的方程组中有 m 个方程, 则使用的发生器定义的样本空间中含 $\text{poly}(m)$ 个 m 比特串, 考虑输入方程相应的线性组合。注意: 只需将发生器的偏移量限定为 $1/6$, 并使用一个 ε -偏移的发生器便可得到一个将 $1/3$ 用 $0.5 - \varepsilon$ 代替时的类似结果。

10.7 (通过扩张图实现多向(Multi-way)不等式) 本题的目的是提出一种可用于设计近似问题间归约的技术(Papadimitriou 和 Yannakakis^[179])。鉴于 $\text{gapSAT}_{0,1}^3$ 是 NP-困难的, 我们的目标是证明如下缝隙问题的 NP-困难性, 该问题记作 $\text{gapSAT}_{\varepsilon}^{3,c}$, 是 $\text{gapSAT}_{\varepsilon}^3$ 的一种特例。其实例被限定为 3CNF 范式, 其中每个变量至多出现在 c 个子句中, c (与 ε 一样) 为给

定常量。注意：3SAT 到相应特例的标准归约（见命题 2.23）并不保持近似缝隙。^①其思想是通过只在对应于扩张图中边的变量对之间引入相等条件来强制性地令辅助变量（即每个原始变量的副本）的赋值相等（见附录 E.2）。例如，为了强制性地令 $z^{(1)}, z^{(2)}, \dots, z^{(m)}$ 的值相等，可以对任意 $\{i, j\} \in E$ ，增加子句 $z^{(i)} \vee \neg z^{(j)}$ ，其中 E 为 m -结点扩张图的边集。证明对于常数 c 及 $\varepsilon > 0$ ，相应的映射将 $\text{gapSAT}_{0,1}^3$ 归约为 $\text{gapSAT}_{\varepsilon}^{3,c}$ 。

提示：在上述归约中使用 d -正则扩张，将普通 3CNF 范式映射为每个变量至多出现于 $2d+1$ 个子句中的 3CNF。注意：增加的子句数量与初始子句数量呈线性关系。显然，如果初始范式是可满足的，则归约后的范式也是可满足的。另一方面，考虑对归约后公式 ϕ' (ϕ 的像) 的任意一种赋值 τ' 。对每个初始变量 z ，如果 τ' 为 z 的几乎所有副本都赋相同的值，则考虑 ϕ 中相应的赋值；否则，由于增加子句的影响， τ' 不会满足包含 z 的副本的子句中占常数比例的部分。

10.8 （多数表决需要线性时间） 证明当使用错误概率不超过 $1/3$ 的随机算法时，多数表决需要线性时间完成，即使在直接访问模型下也是如此。

提示：考虑区分 X_n 与 Y_n ，其中 X_n （或 Y_n ）在由恰含 $\lfloor n/2 \rfloor$ （或 $\lfloor n/2 \rfloor + 1$ ）个 0 的 n 比特串构成的集合中均匀分布。对任意给定的集合 $I \subset [n]$ ，将 X_n （或 Y_n ）在 I 上的投影记作 X'_n （或 Y'_n ）。证明 X'_n 与 Y'_n 的统计距离上限为 $O(|I|/n)$ 。注意：需将这一结论推广到检查的位置为适应性选择的情形。

10.9 （多项式—对数时间内的多数测试） 证明多数测试（在定义 10.11 的意义下）可在多项式—对数时间内完成，方法是检查输入中随机选择的常数个位置。

10.10 （某些测试问题的平凡性） 证明以下集合在邻接矩阵表示下可以平凡地测试（即对任意 $\delta > 0$ ，以及任意这样的集合 S ，存在区分 S 与 $\Gamma_{\delta}(S)$ 的平凡算法）。

- (1) 连通图集合。
- (2) Hamilton 图集合。
- (3) Eulerian 图集合。

事实上，证明对于每种情形，均有 $\Gamma_{\delta}(S) = \phi$ 。

提示（第 (3) 项）：注意到，一般而言，集合 S' 与 S'' 在某个复杂度下是可测试的，这并不蕴含着 $S' \cap S''$ 在相同复杂度下也是可测试的。

10.11 （ tpcP 的等价定义） 证明 $(S, X) \in \text{tpcP}$ 当且仅当存在多项式时间算法 A ，使得 $A(X_n)$ 的错误概率（在判定 S 成员时）是关于 n 的可忽略函数。

10.12 （ tpcP 与 $\mathcal{P}$ ——第一部分） 证明 tpcP 中包含问题 (S, X) 满足 S 不是递归的。进一步地，利用 $X = U$ 。

提示：令 $S = \{0^{|x|} : x \in S'\}$ ，其中 S' 是任意一个（非递归）集合。

^① 回顾在这种归约下，任一布尔变量在每次出现时，均被该变量一个新的副本所代替，而为了使所有副本赋值相同，需增加一些子句。具体而言，变量 z 的 m 次出现用变量 $z^{(1)}, z^{(2)}, \dots, z^{(m)}$ 所代替，并增加子句 $z^{(i)} \vee \neg z^{(i+1)}$ 及 $z^{(i+1)} \vee \neg z^{(i)}$ ($i=1, 2, \dots, m-1$)。问题在于，在归约后的公式中，几乎所有子句都可被一种赋值所满足，其中每个变量的一半副本赋值相同，其余副本的赋值则与之相反。也就是说，如果一种赋值使 $z^{(1)} = \dots = z^{(i)} \neq z^{(i+1)} = \dots = z^{(m)}$ ，则为了令 z 的所有副本相等而引入的辅助子句中，只有一个是不能满足的。如果使用另一种归约，其中对任意 $\{i, j\} \in [m]$ ，增加子句 $z^{(i)} \vee \neg z^{(j)}$ ，则上述结论不再成立，因为增加的子句数量可能是初始子句数量的平方。

10.13 (tpcP与P——第二部分) 证明存在分布问题 (S, X) 满足 $S \notin P$, 且仍存在一个解 S 的算法(对所有输入均能正确求解), 运行时间是关于 X 的多项式。进一步地, 利用 $X=U$ 。

提示: 对任意可定时构造的超多项式及亚指数的函数 $t: \mathbf{N} \rightarrow \mathbf{N}$, 对任意 $S' \in D_{\text{TIME}}(t) \setminus P$, 利用 $S = \{0^{|x|}x : x \in S'\}$ 。

10.14 (简单分布与单调采样) 称概率空间 $X = \{X_n\}_{n \in \mathbf{N}}$ 可利用单调映射在多项式时间内采样, 如果存在多项式 p 及多项式时间可计算的函数 f , 满足以下两个条件:

(1) 对任意 n , 随机变量 $f(U_{p(n)})$ 与 X_n 分布相同。

(2) 对任意 n 及任意 $r' < r'' \in \{0, 1\}^{p(n)}$, 有 $f(r') \leq f(r'')$, 其中不等式针对着串的标准词典序。

证明 X 是简单的当且仅当其可利用单调映射在多项式时间内采样。

提示: 先证必要性。假设 X 是简单的, 令 p 为对输入 x , 输出 $\Pr[X_{|x|} \leq x]$ 的算法运行时间的多项式上限(从而, $\Pr[X_{|x|} \leq x]$ 的二进制表示长度不超过 $p(|x|)$)。当数字 $0.r$ (形式如此)位于区间 $[\Pr[X_n < x], \Pr[X_n \leq x]]$ 时, 定义 $f(r) = x$, 从而得到期望的函数 $f: \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$ 。注意到, 利用 X 为简单分布这一事实, 可通过二分查找法计算 f 。再证充分性, 注意到, 利用二分查找法, 能对任意一个可高效计算的单调映射 $f: \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$ 快速求逆。进一步地, 可利用简单方法来高效地确定映射为任意给定 n 比特串的 $p(n)$ 比特串的区间。

10.15 (保持典型多项式时间可解性质的归约) 证明如果分布问题 (S, X) 可归约为分布问题 (S', X') 且 $(S', X') \in \text{tpcP}$, 则 (S, X) 也属于 tpcP 。

提示: 令 B' 为分布问题 (S', X') 的异常实例集, 即对 B' 中实例求解的算法或者出错, 或者运行超时。证明 $\Pr[Q(X_n) \cap B' \neq \emptyset]$ 是一个(关于 n 的)可忽略函数, 可利用 $\Pr[y \in Q(X_n)] \leq p(|y|) \cdot \Pr[X_{|y|}' = y]$ 以及对任意 $y \in Q(x)$, $|x| \leq p'(|y|)$ 。具体地, 利用后一个条件, 可以推断出 $\sum_{y \in B'} \Pr[y \in Q(X_n)]$ 等于 $\sum_{y \in \{y' \in B' : p'(|y'|) \geq n\}} \Pr[y \in Q(X_n)]$, 其上限为 $\sum_{m: p'(m) \geq n} p(m) \cdot \Pr[X_m' \in B']$ (这又是关于 n 可忽略的)。

10.16 (保持无错可解性质的归约) 续习题 10.15, 证明该归约可保持无错可解性(即求解算法从不出错且运行于多项式时间)。

10.17 (归约的传递性) 证明分布问题(如定义 10.16)间的归约具有传递性。

提示: 要点是证明复合归约的支配性。可借助归约从不发出过短的询问这一假设。

10.18 对任意 $S \in \mathcal{NP}$, 构造一个简单概率空间 X , 使得将定理 2.19 证明中使用的一般归约应用于 (S, X) 时, 不满足关于 (S_u, U') 的支配性条件。

提示: 考虑 $X = \{X_n\}_{n \in \mathbf{N}}$, 使得 X_n 在 $\{0^{n/2}x' : x' \in \{0, 1\}^{n/2}\}$ 上均匀分布。

10.19 (编码引理的变形) 证明编码引理(见定理 10.17 的证明)的以下两种变形。

(1) 一种变形针对着任意可高效计算的函数 $\mu: \{0, 1\}^* \rightarrow [0, 1]$, 且对任意 $x' < x'' \in \{0, 1\}^*$,

函数在 $\{0,1\}^*$ 上是单调非增的 (即对任意 $x' < x'' \in \{0,1\}^*$, 有 $\mu(x') \leq \mu(x'')$)。也就是说, 与定理 10.17 的证明不同, 这里对任意 n , 有 $\mu(0^{n+1}) \geq \mu(1^n)$ 。

(2) 与第一部分相同, 只是在这种变形中, 函数 μ 是严格递增的, 且压缩条件要求 $|C_\mu(x)| \leq \log_2(1/\mu'(x))$, 而不是 $|C_\mu(x)| \leq 1 + \min\{|x|, \log_2(1/\mu'(x))\}$, 其中 $\mu'(x) \stackrel{\text{def}}{=} \mu(x) - \mu(x-1)$ 。

对这两种情况的证明都不如本章正文中的证明那样复杂。

10.20 证明对 distNP 中任意问题 (S, X) , 存在一种简单概率分布 Y , 利用在定理 2.19 证明中的归约可将 (S, X) 归约到 (S_Y, Y) 。

提示: 考虑 $Y = \{Y_n\}_{n \in \mathbb{N}}$, 使得 Y_n 为实例 $\langle M, x, l' \rangle$ 的赋值是一个与 $\pi_x \stackrel{\text{def}}{=} \Pr[X_{|x|} = x]$ 成正比的概率。具体地, 对任意 $\langle M, x, l' \rangle$, 有 $\Pr[Y_n = \langle M, x, l' \rangle] = 2^{-|M|} \cdot \pi_x / \binom{n}{2}$, 其中 $n \stackrel{\text{def}}{=} |\langle M, x, l' \rangle| = |M| + |x| + t$ 。另外, 当 $M = M_S$, $t = p_S(|x|)$ 时, 可令 $\Pr[Y_n = \langle M, x, l' \rangle] = \pi_x$, 否则, 令 $\Pr[Y_n = \langle M, x, l' \rangle] = 0$, 其中 M_S 和 P_S 的定义如定理 2.19。

10.21 (单调可标记性 (Markability) 及单调归约) 续习题 2.30, 称集合 T 是单调可标记的, 如果存在多项式时间标记算法 M , 满足

(1) 对任意 $z, \alpha \in \{0,1\}^*$, $M(z, \alpha) \in T$ 当且仅当 $z \in T$ 。

(2) 单调性: 对任意 $|z'| = |z|$ 及 $\alpha' < \alpha''$, 有 $M(z', \alpha') < M(z'', \alpha'')$, 其中不等式针对着串的标准词典序。

(3) 附加的长度要求:

① 如果 $|z'| = |z|$ 且 $|\alpha'| = |\alpha|$, 则 $|M(z', \alpha')| = |M(z'', \alpha'')|$ 。

② 如果 $|z'| \leq |z|$ 且 $|\alpha'| < |\alpha|$, 则 $|M(z', \alpha')| < |M(z'', \alpha'')|$ 。

③ 存在一个 1-1 多项式 $p: \mathbb{N} \rightarrow \mathbb{N}$, 使得对任意 l 及任意 $z \in \bigcup_{i=1}^l \{0,1\}^i$, 存在 $t \in [p(l)]$, 满足 $|M(z, l')| = p(l)$ 。

前两个要求 (条件 (3) 中) 蕴含了 $|M(z, \alpha)|$ 是 $|z|$ 和 $|\alpha|$ 的一个函数, 且随 α 的增加而增加。第三个要求蕴含了对任意 l , 任意长度不超过 l 的串可以映射到长为 $p(l)$ 的串。

注意: 条件 (1) 是习题 2.30 的翻版, 而条件 (2) 和条件 (3) 是新条件。证明如果集合 S 可以 Karp-归约为集合 T , 而 T 是单调可标记的, 则 S 到 T 的 Karp-归约是单调且长度不变的 (即归约满足命题 10.18 的条件)。

提示: 给定一个 S 到 T 的 Karp-归约 f , 首先得到一个 S 到 T 的长度不变归约 f' (将标记算法应用于 $f(x)$, 同时利用条件 (1) 及 ③)。特别地, 可以保证如果 $|x'| > |x''|$, 则 $|f'(x')| > |f'(x'')|$ 。下一步, 得到同样为单调的归约 f'' (如可令 $f''(x) = M(f'(x), x)$, 并利用条件 (1) 和条件 (2))。①

① 实际上, 条件 (2) (与 f' 的长度不变性相结合) 只涉及到关于等长串的单调性。为了保证关于不同长度串的单调性, 我们还使用条件 ② (以及 $|f'(x')| > |f'(x'')|$ 对 $|x'| > |x''|$)。

10.22 (单调可标记性与可标记性) 证明如果一个集合是单调可标记的(如习题 10.21), 则它是可标记的(如习题 2.30)。

提示: 令 M 表示单调标记算法。对于初学者, 可以假设 M 为双射, 并定义 $M'(z, \alpha) = M(z, (z, \alpha))$ 。注意: 原像 (z, α) 可用二分查找法(对所有可能的 $|z|$ 值)求得。一般情况下, 可以修改构造以保证 M' 为双射。具体地, 令 $\text{idx}(n, m) = n + \sum_{i=2}^{n+m} (i-1)$ 为 (n, m) 在枚举所有正整数对时的索引, p 如条件③。然后令 $M'(z, \alpha) = M\left(z, C_{t(|z|, |\alpha|)}(\langle z, \alpha \rangle)\right)$, 其中 $t(n, m) = \omega(n+m)$ 满足 $\left|M(1^n, 1^{t(n, m)})\right| = p(\text{idx}(n, m))$, 且 $C_t(y)$ 表示 y 利用 t -比特串的单调编码。

10.23 (一些单调可标记的集合) 参考习题 10.21, Karp 关于 NP-完全性的第一篇论文^[138]中提出了 21 个 NP-完全问题, 验证其中每个问题都是单调可标记的。对于初学者, 考虑 SAT 集合、团及 3-可着色性。

提示: 对于 SAT 问题, 考虑如下标记算法 M 。算法使用两个(固定的)等长的可满足性公式, 记作 ψ_0 和 ψ_1 , 且满足 $\psi_0 < \psi_1$ 。对任意布尔公式 ϕ 及任意二进制串 $\sigma_1 \sigma_2 \cdots \sigma_m \in \{0, 1\}^m$, 有 $M(\phi, \sigma_1 \sigma_2 \cdots \sigma_m) = \psi_{\sigma_1} \wedge \cdots \wedge \psi_{\sigma_m} \wedge \phi$, 其中 ψ_0 和 ψ_1 中的变量不出现在 ϕ 中。注意: ψ_σ 的多次出现也易避免(通过使用 ψ_σ 的“变形”)。

10.24 (随机归约) 根据 10.2.1.3 节中的概述, 给出分布问题间随机归约的定义。

(1) 类似于习题 10.15, 证明随机归约保持可行的可解性(即在概率多项式时间内典型地可求解)。也就是说, 如果分布问题 (S, X) 可以随机地归约到分布问题 (S', X') , 且 $(S', X') \in \text{tpcBPP}$, 则 (S, X) 也属于 tpcBPP 。

(2) 类似于习题 10.16, 证明随机归约保持概率算法的可解性, 其对每个输入的出错概率至多为 $1/3$ 且典型地运行于多项式时间。

(3) 证明随机归约具有传递性(见习题 10.17)。

(4) 证明随机归约的错误概率可以减少(同时保持支配性条件)。

将上述归约推广到分布式搜索问题中。

10.25 (简单与可采样的概率空间——第一部分) 证明任意简单概率空间可在多项式时间内采样。

提示: 见习题 10.14。

10.26 (简单与可采样的概率空间——第二部分) 假设 $\#P$ 中包含了不可在多项式时间内计算的函数, 证明存在并非简单的多项式时间可采样概率空间。

提示: 考虑任意 $R \in \text{PC}$, 并假设 p 是一个多项式, 满足 $(x, y) \in R$ 蕴含了 $|y| = p(|x|)$ 。考虑采样算法 A , 对于输入 1^n , 均匀选择 $(x, y) \in \{0, 1\}^{n-1} \times \{0, 1\}^{p(n-1)}$, 并当 $(x, y) \in R$ 时输出 $x1$, 否则, 输出 $x0$ 。注意: $\#R(x) = 1^{|x|+p(|x|)} \cdot \Pr[A(1^{|x|+1}) = x1]$ 。

10.27 (NPC 问题的分布式版本——第一部分[30]) 证明如果 S_u 可以 Karp-归约到 S , 且归约过程不压缩输入, 则存在一个多项式时间可采样的概率空间 X , 使得 distNP 中任意问题可归约为 (S, X) 。

提示: 证明对某个可采样的概率空间 X , S_u 到 S 的归约也可将 (S_u, U') 归约为 (S, X) 。考虑第一种情况, 即 S_u 到 S 的标准归约是保持长度的, 并证明应用于可采样概率空间时, 该归

约诱导出 S 实例集上的一种可采样分布 (注意: U' 是可采样的 (由习题 10.25))。然后将结论推广到一般情况, 其中利用到 U'_n 的标准归约可诱导出 $\bigcup_{m=n}^{\text{poly}(n)} \{0,1\}^m$ 上的分布 (而非 $\{0,1\}^n$ 上的分布)。

10.28 (NPC 问题的分布式版本——第二部分^[30]) 证明定理 10.25 (即如果 S_u 可以 Karp-归约为 S , 且归约过程不压缩输入, 则存在一个多项式时间可采样的概率空间 X , 使得 $\text{samp}\mathcal{NP}$ 中任意问题可归约到 (S, X))。

提示: 对于 $S = S_u$ 证明断言成立。再利用从 S_u 到 S 的归约来证明一般情况 (如习题 10.27)。从而, 重点在于要证明, 对某个 (适当选择的) 可采样的概率空间 X , 任意 $(S', X') \in \text{samp}\mathcal{NP}$ 可归约为 (S_u, X) 。不严格地说, X 可以是关于所有可采样分布的一种充分的凸组合 (从而 X 既不等于 U' 也不是简单的)。具体而言, $X = \{X_n\}_{n \in \mathbb{N}}$ 被定义为 X_n 的采样算法均匀选择 $i \in [n]$, 模仿第 i 个 (依词典序) 算法对于输入 1^n 的运行 n^3 步, ^① 并输出后者的输出 (或者当该算法在 n^3 步之内不停机时输出 0ⁿ)。证明对任意 $(S'', X'') \in \text{samp}\mathcal{NP}$, 且 X'' 可在立方时间内采样, S'' 到 S_u 的标准归约将 (S'', X'') 归约为 (S_u, X) (如定义 10.15, 即特别地, 满足支配性条件)^②。最后, 利用充分的填充将任意 $(S', X') \in \text{samp}\mathcal{NP}$ 归约为某个 $(S'', X'') \in \text{samp}\mathcal{NP}$, 使得 X'' 可在立方时间内采样。

10.29 (可采样空间中的搜索与判定问题) 证明 $\text{samp}\mathcal{NP}$ 中任意问题可归约为 $\text{samp}\mathcal{PC}$ 中的某个问题, 而 $\text{samp}\mathcal{PC}$ 中任意问题可以随机地归约为 $\text{samp}\mathcal{NP}$ 中的某个问题。

提示: 见定理 10.23 的证明。

① 当然, 考虑 n 个算法 (在 X_n 的定义中) 是相当随意的。任意关于 n 的至多为多项式的无界限函数 (且在多项式时间内可计算) 都可以 (更一般地, 可以选择第 i 个算法为 p_i , 只要 p_i 是关于 n 的不可忽略函数即可)。类似地, 模仿每个算法的步骤数为 n^3 (而不是其他的多项式数量的步骤) 也相当随意。

② 注意: 利用到 X'' 的这种归约, 可以得到一个能在立方时间内采样的空间。这一断言利用了一个事实, 即标准归约的运行时间小于其输出的立方时间 (通常总是运行于线性时间), 并且输出总是比输入长。

附录 A

复杂性类汇总

内容提要

本附录包括了本书中提及的大部分复杂性类的自包含的定义。当然，对这些类具体讨论得比较少，读者要想更详细地了解请看正文。词条是根据讨论的问题来组织的，而不是根据字母顺序。特别地，术语可分两部分：依据算法和资源（即图灵机的时间和空间复杂性）定义的复杂性类以及依据非一致性电路（及电路规模和深度）定义的复杂性类。算法类包括时间复杂性类（如 P 、 NP 、 $coNP$ 、 BPP 、 RP 、 $coRP$ 、 PH 、 E 、 EXP 和 $NEXP$ ）和空间复杂性等级（ L 、 NL 、 RL 和 $PSPACE$ ）非一致性类包括电路类 $P/poly$ ，以及 NC^k 和 AC^k 。

网站 Complexity Zoo^[1]上可以看到许多其他复杂性类的定义（及结论）。

A.1 预备知识

复杂性类实际上是一些计算问题的集合，每个类中包含可以用特定计算资源解决的问题。在定义复杂性类时，需要规定计算模型、作为输入长度函数的复杂性度量（如时间或空间）以及（该类中问题的）复杂度界限。

我们照例重点讨论判定问题，但却利用了承诺问题（见 2.4.1 节）的术语。也就是说，把 Π_{no} 中的输入与 Π_{yes} 中的输入区分开来，用 (Π_{yes}, Π_{no}) 来表示相应的判定问题。标准的判定问题被视为 $\Pi_{yes} \cup \Pi_{no} = \{0,1\}^*$ 的特殊情况，此时是一种平凡承诺。

图灵机是一种流行的计算模型。该模型使用（一致性）算法（见 1.2.3 节）的概念。另一个重要的模型是非一致性的电路（见 1.2.4 节）。术语“非一致性”是指算法对每种输入长度而言是否为同一个算法（或电路）。

我们主要考虑在一般的复杂性量度和限制下得到的复杂性类。典型地，这些类中包括自然的计算性问题（定义见附录 G）。而且，几乎所有的类中都包含完全问题，即问题属于该类，且类中的每个问题都可“易于”归约到这些问题。“易于”是指归约所需资源比求解该类中单个问题所需资源要少。因此，求解完全问题的高效算法中蕴含了求解该类中所有问题的高效算法。

内容组织

本附录按专题来组织（而非字母顺序）。特别地，我们区分了根据算法资源（如图灵机的时间和空间复杂性）定义的复杂性类和根据不同的电路（规模和深度）定义的复杂性类。前

者（基于算法的）见 A.2，后者（基于电路的）见 A.3。

A.2 基于算法的类

（一致的）算法的两个主要复杂度指标为算法运行的步数（即时间复杂性）和计算所使用的“内存”或“存储空间”（即它的空间复杂度）。我们回顾基于时间复杂性的类有 $\mathcal{P}$ 、 $\mathcal{NP}$ 、 $\text{co}\mathcal{NP}$ 、 $\mathcal{BPP}$ 、 $\mathcal{RP}$ 、 $\text{co}\mathcal{RP}$ 、 $\mathcal{ZPP}$ 、 $\mathcal{PH}$ 、 $\mathcal{E}$ 、 $\mathcal{EXP}$ 和 $\mathcal{NEXP}$ ，基于空间复杂性的类有 $\mathcal{L}$ 、 $\mathcal{NL}$ 、 $\mathcal{RL}$ 和 $\mathcal{PSPACE}$ 。

顾名思义，具有前缀“co”的复杂性类（判定性问题）表示补问题；也就是说，当且仅当 $(\Pi_{\text{no}}, \Pi_{\text{yes}})$ 在 C 中时，问题 $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ 在 $\text{co}C$ 中。特别是，当且仅当集合 $\{0,1\}^*S$ 中的成员判定性问题属于 C 类时，集合 S 中的成员判定性问题属于 $\text{co}C$ 类。于是， $\text{co}\mathcal{NP}$ 和 $\text{co}\mathcal{RP}$ 的定义可以很容易地分别从 $\mathcal{NP}$ 和 $\mathcal{RP}$ 的定义推导出来。依据对称可接受准则（Symmetric Acceptance Criteria）定义的复杂性类（如确定性的双边错误随机的类）在补集下都是封闭的（如， $\text{co}\mathcal{P} = \mathcal{P}$ ， $\text{co}\mathcal{BPP} = \mathcal{BPP}$ ），因此没有给出其“co”类。其他情况下（尤其是 $\mathcal{NL}$ ）的封闭性则不一定成立，我们将对此进行说明。

A.2.1 时间复杂性类

从与多项式时间算法（如 $\mathcal{P}$ 、 $\mathcal{NP}$ 、 $\mathcal{BPP}$ 、 $\mathcal{RP}$ 和 $\mathcal{ZPP}$ ）密切相关的类开始，后面将考虑 $\mathcal{PH}$ 、 $\mathcal{E}$ 、 $\mathcal{EXP}$ 和 $\mathcal{NEXP}$ 类。

A.2.1.1 与多项式时间密切相关的类

最常见的复杂性类是 2.1 节深入讨论的 $\mathcal{P}$ 和 $\mathcal{NP}$ 。6.1 节还讨论了与随机多项式时间相关的类。

$\mathcal{P}$ 和 $\mathcal{NP}$

$\mathcal{P}$ 类包含所有可在多项式时间内解决的（确定性的）判定问题。如果存在一个多项式 p 和一个（确定性的）多项式时间算法 V 使得以下两个条件成立，则判定性问题 $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ 属于 $\mathcal{NP}$ ：

- (1) 对任意的 $x \in \Pi_{\text{yes}}$ ，存在 $y \in \{0,1\}^{p(|x|)}$ ，使得 $V(x,y)=1$ 。
- (2) 对任意的 $x \in \Pi_{\text{no}}$ 和 $y \in \{0,1\}^*$ ，有 $V(x,y)=0$ 。

满足条件 (1) 的字符串 y 称为一个（对 x 的）NP-证据，显然，有 $\mathcal{P} \subseteq \mathcal{NP}$ 。

归约和 NP 完全问题（NPC）

如果一个问题属于 $\mathcal{NP}$ ，并且每个属于 $\mathcal{NP}$ 的问题都可以在多项式时间内归约到该问题，那么该问题就是 $\mathcal{NP}$ -完全问题，这里多项式时间可归约性的定义见 2.2 节。粗略地讲，从问题 Π 到问题 Π' 的多项式时间归约，是一个通过询问解决问题 Π' 的子程序来解决问题 Π 的多项式时间算法，这里子程序的运行时间不计入算法的时间复杂度。典型地，NP-完全问题是在将归约限制为发出单个询问并输出其应答的情况下定义的。这样的归约称为 Karp-归约，由多项式时间可计算的映射来表示，该映射将 Π 的“yes”实例映射到 Π' 的“yes”实例（和 Π 的“no”实例到 Π' 的“no”实例）。参考文献[85]中列出了几百个 NP-完全问题。

概率多项式时间 (BPP、RP 和 ZPP)

如果存在一个概率多项式时间算法 A 满足下面两种情况, 则称判定性问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$ 属于 **BPP**:

- (1) 对任意的 $x \in \Pi_{\text{yes}}$, 有 $\Pr[A(x)=1] \geq 2/3$ 。
- (2) 对任意的 $x \in \Pi_{\text{no}}$, 有 $\Pr[A(x)=0] \geq 2/3$ 。

该算法有双边错误概率 (为 $1/3$), 它可以通过迭代来减少。要强调的是, 由于 **BPP** 具有双边错误概率, 因而不知道 **BPP** 是否包含在 **NP** 中。除了双边错误类 **BPP**, 我们还考虑了分别表示为 **RP** 和 **ZPP** 的单边错误和零错误类。如果存在一个概率多项式时间算法 A 满足下面两个条件, 则问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$ 属于 **RP**:

- (1) 对任意的 $x \in \Pi_{\text{yes}}$, 满足 $\Pr[A(x)=1] \geq 1/2$ 。
- (2) 对任意的 $x \in \Pi_{\text{no}}$, 满足 $\Pr[A(x)=0]=1$ 。

依然可以通过重复来减少错误概率, 于是, $\text{RP} \subseteq \text{BPP} \cap \text{NP}$ 。如果存在一个可能性多项式时间算法 A , 可能输出一个特殊 (“不知道的”) 符号 $\perp$, 并且满足下面两个条件, 则问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$ 属于 **ZPP**:

- (1) 对任意的 $x \in \Pi_{\text{yes}}$, 满足 $\Pr[A(x) \in \{1, \perp\}=1]=1$ 和 $\Pr[A(x)=1] \geq 1/2$ 。
- (2) 对任意的 $x \in \Pi_{\text{no}}$, 满足 $\Pr[A(x) \in \{0, \perp\}=1]=1$ 和 $\Pr[A(x)=0] \geq 1/2$ 。

注意到, $\text{P} \subseteq \text{ZPP} = \text{RP} \cap \text{coRP}$ 。当使用承诺问题术语进行定义时, 所有之前提到的随机类都存在完全问题 (关于 Karp-归约), 但当仅考虑标准判定性问题时同样是未知的 (通过平凡的承诺)。

计数类 #P

#P 中的函数对 **NP**-类型搜索问题的解的数量 (或者相当于 **NP** 中判定问题的 “是” 实例的 **NP** 证据的数量) 进行计数。严格地说, 如果存在多项式 p 和一个 (确定性) 多项式时间算法 V 满足 $f(x)=|\{y \in \{0, 1\}^{p(|x|)}: V(x, y)=1\}|$, 则函数 f 属于 **#P**。 p 和 V 其实与 **NP** 中的定义一样, 从而判定集合 $\{x: f(x) \geq 1\}$ 的成员属于 **NP**。另外, **#P** 问题在多项式空间中一定是可解的。令人惊奇的是, 求正整数矩阵的常值是 **#P**-完全的 (即属于 **#P**, 且 **#P** 中任何函数都可以在多项式时间内归约为该问题)。

交互式证明

如果存在一个多项式时间策略 V 满足如下两个条件, 则称判定问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$ 有交互式证明系统:

- (1) 对任意的 $x \in \Pi_{\text{yes}}$ 存在一个证明策略 P , 使得在输入公共参数 x 时, 经过与证明者 P 的交互后, 验证者 V 总是接受。
- (2) 对任意的 $x \notin \Pi_{\text{no}}$ 和任意策略 P^* , 在输入公共参数 x , 经过与 P^* 交互后, 验证者 V 至少以 $1/2$ 的概率拒绝。

相应的类记作 **IP**, 它等价于 **PSPACE**。(详见 9.1 节。)

A.2.1.2 其他的时间复杂性类

对 n 比特长的输入来说, 类 **E** 和 **EXP** 对应的问题分别可以在时间 $2^{O(n)}$ 和 $2^{\text{poly}(n)}$ 内解决 (通过确定性的算法)。显然, $\text{NP} \subseteq \text{EXP}$ 。同时, 也提到了 **NEXP**, 它是指可以利用非确定型

图灵机在 $2^{\text{poly}(n)}$ 步内求解的问题类^①。

一般来说, 对任意时间界和任意类型的图灵机 (即确定性的、概率性的和非确定性的) 都可以定义一个复杂度类, 但是多项式和指数界似乎是最自然的也是最健壮的。另一种时间界的健壮类型有时称为准多项式时间 (即 $\tilde{p}$, 表示由时间复杂度为 $\exp(\text{poly}(\log n))$ 的确定性图灵机可解决的问题类)。

多项式时间层级 $\mathcal{PH}$

对任意自然数 k , 多项式时间层级的第 k 级包含问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$, 使得存在一个多项式 p 和一个多项式时间算法 V 满足如下两个条件:

(1) 对任意的 $x \in \Pi_{\text{yes}}$, 存在 $y_1 \in \{0,1\}^{p(|x|)}$, 使得对任意的 $y_2 \in \{0,1\}^{p(|x|)}$ 存在 $y_3 \in \{0,1\}^{p(|x|)}$ 满足任意的 $y_4 \in \{0,1\}^{p(|x|)} \dots$, 满足 $V(x, y_1, y_2, y_3, y_4, \dots, y_k)=1$ 。也就是关于 x 的条件包含 k 个交替的量词。

(2) 对任意的 $x \in \Pi_{\text{no}}$, 前述的 (k 阶交替的) 条件不满足。也就是对任意的 $y_1 \in \{0,1\}^{p(|x|)}$, 使得对任意的 $y_2 \in \{0,1\}^{p(|x|)}$ 存在 $y_3 \in \{0,1\}^{p(|x|)}$ 满足任意的 $y_4 \in \{0,1\}^{p(|x|)} \dots$, 满足 $V(x, y_1, y_2, y_3, y_4, \dots, y_k)=0$ 。

这样一个问题 Π 被称为在 Σ_k (且 $\Pi_k \stackrel{\text{def}}{=} \text{co}\Sigma_k$)。事实上, $\mathcal{NP}=\Sigma_1$ 相当于 $k=1$ 的特殊情况。有趣的是 $\mathcal{PH}$ 可在多项式时间内归约为 $\#P$ 。

A.2.2 空间复杂性类

空间复杂度的定义中只包含实际计算使用的空间, 而不包括输入和输出占用的空间。只有当输入是从只读设备中读取时才能这样定义 (相应地, 输出写在只写设备上)。以下定义了 4 个重要的判定性问题类。

- $\mathcal{L}$ 类包含在对数空间内可以求解的问题。也就是说, 如果存在一个求解 Π 的对数空间复杂度 (即确定性的) 算法, 则问题 Π 在 $\mathcal{L}$ 中。该类包括一些简单的计算问题 (如矩阵乘法), 并且也可以解释空间最高效的计算。有趣的是, $\mathcal{L}$ 包含判定 (无向) 图的连接性问题。

- $\mathcal{RL}$ 类和 $\mathcal{BPL}$ 类包含了用对数空间复杂度随机算法可求解的问题, 它们可仿照 $\mathcal{RP}$ 和 $\mathcal{BPP}$ 来定义。也就是说, $\mathcal{RL}$ 对应单边错误概率的算法, 而 $\mathcal{BPL}$ 允许双边错误。

- $\mathcal{NL}$ 类是 $\mathcal{L}$ 类的非确定性模拟, 习惯上用对数空间复杂度的非确定型机器定义。^② $\mathcal{NL}$ 类包括了判定在给定的有向图中两个给定顶点间是否存在有向路径的问题。事实上, 在该类中后一个问题是完全的 (在对数空间归约下)。有趣的是, $\text{co}\mathcal{NL}$ 等于 $\mathcal{NL}$ 。

- $\mathcal{PSPACE}$ 类由多项空间可求解的问题组成。该类中包含了非常困难的问题, 包括对任意 “高效 2 方游戏” (见 5.4 节) 计算获胜策略。

显然, $\mathcal{L} \subseteq \mathcal{RL} \subseteq \mathcal{NL} \subseteq \mathcal{P}$ 且 $\mathcal{NP} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXP}$ 。

① 也可以仿照 $\mathcal{NP}$ 来给出一个可替换的定义, 如果存在一个多项式 p 和多项式时间算法 V 使得如下两个条件能够成立, 则问题 $\Pi=(\Pi_{\text{yes}}, \Pi_{\text{no}})$ 属于 $\mathcal{NEXP}$:

(1) 对每个 $x \in \Pi_{\text{yes}}$ 存在 $y \in \{0,1\}^{2^{\text{poly}(|x|)}}$ 使存在, 使得 $V(x, y)=1$ 。

(2) 对每个 $x \in \Pi_{\text{no}}$ 和每个 $y \in \{0,1\}^*$ 有 $V(x, y)=0$ 。

② 对该定义进一步的讨论参见 5.3 节。

A.3 基于电路的类

读者可以参考 1.2.4 节中作为计算设备的布尔电路。电路的两个主要的复杂度指标是电路中门（或电线）的数量（即电路的规模）和从输入到输出的最长有向路径的长度（即电路的深度）。

本节中的电路实际上是指包含了每个常数长度电路的电路族，其中计算问题的长为 n 比特的实例由电路族中第 n 个电路来处理。类似地，电路的规模和深度实际上涉及电路族与电路族中第 n 个电路的（与 n 有关的）规模及深度。

一般的多项式规模电路 ($P/poly$)

介绍基于（非一致性）电路的复杂性类的动机是为了寻求下界。例如，用 $P/poly$ 表示由多项式规模的电路可解决的问题类，这是 P 的一个超集。^①于是，证明 NP 不包含于 $P/poly$ 就蕴含了 $P \neq NP$ 。进一步的讨论可参考附录 B.2。在 3.1.2 节中用“接受建议的机器”术语给出了 $P/poly$ 的另一种定义。同时，也提到如果 $NP \subset P/poly$ ，则 $PH = \Sigma_2$ 。

AC^0 和 TC^0 子类

附录 B.2.3 节中的类 AC^0 ，由无界扇入的常数深度多项式规模电路可解的问题组成。用 TC^0 表示也允许（无界扇入的）主要门（Majority-gates）（或等价于门限门 Threshold-gates）模拟类。

AC 和 NC 子类

再回到（ $\neg$ 、 $\vee$ 和 $\wedge$ 的）标准基上来，对任意非负整数 k ，用 NC^k (AC^k) 来表示有界扇入的（无界扇入的）、深度为 $O(\log^k n)$ 的多项式规模电路可解的问题，其中 n 为输入长度。显然， $NC^k \subseteq AC^k \subseteq NC^{k+1}$ 。一个经常被引用的类是 $NC \stackrel{\text{def}}{=} \bigcup_{k \in \mathbb{N}} NC^k$ 。

类 $NC^2 \supseteq NL$ 是线性代数中大多数自然问题的依据：求解一个线性方程组以及求矩阵的秩、逆矩阵和矩阵的行列式。类 NC^1 包含了有限群和幺半群的所有的对称函数、正则语言以及字问题。类 AC^0 包含了由一阶逻辑可表示的有限对象的所有属性。

一致性

以上的类中没有提及构造电路的复杂性，似乎没有高效的方法来构造电路（如用电路来简单地解决不可判定性问题一元实例的情况）。构造（多项式规模）电路的最小概念就是这样一个多项式时间算法的存在性问题，用该算法对给定 1^n 产生第 n 个相关电路（即解决长度为 n 的问题实例的电路）。这个可构造性的概念指在某种意义上这种电路族是“一致的”（而不是包含一个和另一个之间没有关系电路）。关于一致性更强的概念（如对数空间可构造性）对 AC 和 NC 这样的子类是更适当的。对数空间一致的 NC 电路与使用多项式数量的处理器并且运行于多项式时间的并行算法相对应。

^① 特别地， $P/poly$ 中包含一些无法用一致算法求解的判定问题。

附录 B

寻求下限

唉！我绞尽脑汁把哲学、法学和医学，天哪，还有神学，都研究透了。这时的我，这个蠢货！尽管满腹经纶，也并不比以前聪明。称什么学士，称什么博士，十年来牵着我的学生们的鼻子，天南地北，海阔天空，处处驰骋，这才知道我们什么也不懂！^①

J.W.歌德，浮士德，第 354 行~第 364 行

内容提要

本附录概述对自然计算问题的复杂性下界进行的一些证明。第一部分主要围绕电路复杂性，描述了求解自然计算问题的（受限制的）电路规模的下界。这可被视为一个以证明 $P \neq NP$ 为长远目标的计划。第二部分主要围绕证明复杂性，描述对自然重言式的（受限制的）命题证明的长度下界。这可被视为以证明 $NP \neq coNP$ 为长远目标的计划。

目前，该领域内比较流行开发能用于“大问题”（如 P-vs-NP）求解的证明技术，但现有成果（尽管很深刻）似乎离这个目标还很远。该领域最杰出的成果是在受限计算模型（证明系统）下，计算（证明）“相对简单”函数紧致的（或强的）复杂性下界。

B.1 预备知识

电路复杂性涉及非一致性的计算模型，特别是布尔电路模型，主要关注这种电路的规模，而忽视了构造电路的复杂性。类似地，证明系统涉及重言式的证明，也主要关注证明的长度，而忽视了产生证明的复杂性。电路和证明都是定义在后面有讨论的有向无环图（Dag）概念之上的有限对象。

有向无环图 $G(V, E)$ 中包含有限的结点集合 V 、一组称为有向边的有序对 $E \subseteq V \times V$ ，其中没有有向环。没有输入边的结点称为有向无环图 G 的输入，没有输出边的结点称为图 G 的输出。对图作一些限制，令其中每个结点的输入边最多为 2。如果每个结点的输出的最大边数为 1，则该有向无环图称为树。最后，假定输入通过有向路径可达每个结点。有向无环图的规模为其边数。

为将有向无环图转化为计算设备（或证明），将每个内部结点利用某种规则标记，将其前

^① 这句话反映了一种普遍心态，但本书作者并不这样认为。

驱的值转换为该结点的值。与对输入的任意可能赋值相结合，这些固定规则可以为有向无环图上的每个结点赋值（从输入开始，根据其前驱为每个结点赋值（且遵守相应的规则））。

- 对于计算设备，内部结点由某个（有限或无限）域上（二元或一元）的函数来标记。这些函数称为门，标记后的图称为电路。这样一个电路（ n 个输入和 m 个输出）可以计算相应域上的函数（将长为 n 的序列映射到长为 m 的序列）。

- 对于证明，内部结点由某个给定证明系统的推理来标记。对于（该系统中）公理的任意赋值。如果用给定证明系统中的公理来标记图的输入，将得到（在所有结点上的）重言式序列。通常认为，有向无环图有唯一的输出结点，且相应的重言式序列被认为是赋予该输出的证明。

需要指出，这两种类型均符合 1.2 节中所有计算模型共用的简单范式：定义简化了规则——它们被用于至多 2 个前驱值的情况。然而，与计算模型不同的是，这里的模型不把所有可能的“输入”视为等同（而是区别对待每种不同的“输入”长度）。例如，每个电路只能计算一个有限函数，即在有限定义域上的函数（给定输入长度）。类似地，相应于证明系统的有向无环图只能得到针对给定公理数量的重言式的证明。

主要讨论电路，注意到为了计算在所有输入长度上均有定义的函数，必须考虑一个无限的有向无环图序列，每种长度使用一个，从而得到一个拥有无限长度描述（当适用于所有输入长度时）的计算设备模型。这在超越了算法概念的层次上（在 1.2.3 节中讨论）有效地扩展了计算模型的能力。然而，因为我们对下界感兴趣，所以这样做是合法的，也有希望得到新的成果：例如，单个电路的限制便于利用组合技术分析模型的功能和局限性。进一步地，我们将看到，这些模型有助于引入有意义受限的计算类型。

内容组织

本附录分三部分。B.2 讨论布尔电路，这是非一致计算设备的原型。B.3 推广到算术电路，它在每种代数结构上均有定义（而布尔电路只是二元域上的特例）。最后在 B.4 讨论证明的复杂性。

B.2 布尔电路的复杂性

在布尔电路中，为所有输入、中间结点（由计算）诱导出的值以及输出均赋二元值。电路中门的集合构成了一个完全基(Complete Basis)——使电路能够计算所有布尔函数。对完全基的特定选择几乎不会影响电路的复杂性。一种典型的选择是（分别）对与、或（每个对 2 个比特位）和非（对 1 个比特位）的集合 $\{\wedge, \vee, \neg\}$ 。

对于有限函数 f ，用 $S(f)$ 表示计算 f 的最小布尔电路的规模。主要讨论函数序列 $\{f_n\}$ ，其中 f_n 是有 n 个输入比特的函数，并渐近地研究其规模复杂性（即 $S(f_n)$ ，是 n 的函数）。稍微滥用一下符号，对 $f(x) = f_{|x|}(x)$ ，令 $S(f)$ 表示对 n 赋值 $S(f_n)$ 的整数函数。于是，得到如下定义。

定义 B.1(电路复杂性) 令 $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 和 $\{f_n\}$ 对所有 x 满足 $f(x) \stackrel{\text{def}}{=} f_{|x|}(x)$ 。函数 $f(\{f_n\})$ 的复杂性记作 $S(f)$ （记为 $n \mapsto S(f_n)$ ），表示计算 f_n 的最小布尔电路的规模，它是 n 的函数。

注意到，每个 f_n 使用了不同的电路（如拥有不同的输入数）。这个电路序列仍可被简单地描述为，对输入的 n 产生计算 f_n 的电路的算法。如果这种算法存在且其工作时间是其输出规

模的多项式, 则称相应的电路序列是一致的。注意: 如果 f 有一个一致的多项式规模电路序列, 则 $f \in \mathcal{P}$ 。另一方面, 也可以表明, 对任意 $f \in \mathcal{P}$ 都有 (均匀序列) 多项式规模电路。因此, $\mathcal{NP}$ 中任意函数的超多项式规模下界蕴含了 $\mathcal{P} \neq \mathcal{NP}$ 。

定义 B.1 没有涉及到“一致性”, 而且实际上计算 $\{f_n\}$ 的最小电路序列可能是高度“非一致的”。非一致性使得电路模型比图灵机更强大 (或等价地, 比一致的电路模型更强): 存在图灵机计算不了 (不论其运行时间) 但拥有线性规模电路的函数 f 。^① 因此, 证明电路的下界可能是一个比解决 $\mathcal{P}$ vs. $\mathcal{NP}$ 问题更难的任务。

一般认为, 非一致性提供的额外能力与 $\mathcal{P}$ vs. $\mathcal{NP}$ 问题无关, 也就是推测 $\mathcal{NP}$ -完全集不存在多项式规模的电路。有一个事实能支持这个推测, 它不成立将导致标准计算复杂性体系 (见 3.2 节) 的意外崩溃。此外, 人们希望通过抽象来去掉 (可能是无关紧要的) 一致性条件, 这样将得到分析多项式规模电路 (关于 $\mathcal{NP}$ 集) 的功能和局限性的组合技术。在受限的电路类 (见附录 B2.2 和附录 B2.3) 中这种希望可以实现。事实上, 电路模型的另一个作用是为自然限制的计算模型提供了一个框架。

还要指出, 布尔电路是对应于“硬件复杂性” (实际上香农^[203]提出它们的原始动机) 的自然计算模型, 所以对其研究具有独立的意义。布尔函数的分析技术在其他地方也得到了大量应用 (如在计算学习理论, 组合论和博弈论中)。

B.2.1 基本结论和问题

已经讨论了关于布尔电路的几个基本事实, 特别是它们可以高效地仿真图灵机。另一个事实是大部分布尔函数需要指数规模的电路, 原因在于函数数量与小电路数量之间的缝隙。

于是, 至少可以说困难函数 (即需要大电路并且没有高效算法的函数) 确实存在。然而, 上述困难结果是通过计算参数证明的, 无法构造出一个具体的困难函数。更糟糕的是, 对任意显式函数 f , 超线性的电路规模下限尚且未知, 即使以非常温和的形式定义, 只要求 $f \in \mathcal{EXP}$ 时也是如此。^② 一个关于电路复杂性的主要开放问题是打破这个平凡界限。

开放问题 B.2 找到一个显式布尔函数 f (或即便是一个保持长度的函数 f), 使得 $S(f)$ 不等于 $O(n)$ 。

该问题一个基本的特殊情况是, 加法是否比乘法更容易实现。令 $\text{ADD}: \{1, 0\}^n \times \{1, 0\}^n \rightarrow \{1, 0\}^{n+1}$ 以及 $\text{MULT}: \{1, 0\}^n \times \{1, 0\}^n \rightarrow \{1, 0\}^{2n}$, 分别表示应用于一对整数 (以二进制表示) 的加法和乘法函数。对加法已有最优上界, 即 $S(\text{ADD}) = O(n)$ 。对于乘法, 标准的二次时间算法能被大幅提升 (通过离散傅里叶变换) 至近超线性水平, 得到 $S(\text{MULT}_n) = O(n \cdot (\log n)^2)$ 。但问题是对于乘法是否存在线性规模电路 (即 $S(\text{MULT}_n) = O(n)$)。

无法给出 (对显式函数的) 任意非平凡下界, 那么转向受限模型。在证明自然受限电路类的强下界方面, 已成功开发了一些技术。此处, 描述其中最重要的, 进一步细节可见参考文献 [46, 236]。

如上所述的普通布尔电路能够计算每种函数, 并能至少和普通 (一致的) 算法一样完成运算。受限电路可能仅能计算所有函数的一个子类 (如单调函数)。当相关的函数类和由受限

① 见定理 1.13 或定理 3.7。

② 事实上, 一个更自然的 (仍为温和的) 明确性要求 $f \in \mathcal{E}$ 。这一概念暗示了函数的描述 (限制为 n 比特输入) 可在其长度的多项式时间内构造。

电路所表达的计算为正常计算时（从概念或实践的角度来看），这种限制就是合理的。以下模型满足这个条件。

B.2.2 单调电路

通过在门中禁止取反，就是仅允许 $\{\wedge, \vee\}$ ，可以得到一个极自然的约束。由此得到的电路称为单调电路，而且显然此类电路能够计算对 n 比特字符的标准部分顺序单调的所有函数 $f: \{0, 1\}^n \rightarrow \{0, 1\}$ （即如果在每个比特位置 i 都有 $x_i \leq y_i$ ， $x \leq y$ ）。这里一个很自然的问题是：（在电路中的）非单调操作在计算单调函数时是否有用？

在讨论该问题之前，要指出显然绝大多数单调函数需要指数规模的电路（更不必说单调的）^①。证明显式单调函数的单调电路复杂性的超多项式下界仍然是一个存在了 40 多年的开放问题，直到发明了所谓的近似方法（Approximation Method，由 Razborov^[187]提出）。

函数 CLIQUE 定义为：给定 n 个结点的图（由其邻接矩阵给出），当且仅当图中包含一个规模为 $\sqrt{n}$ 的完全子图时（即在大约 $\sqrt{n}$ 的子集中的所有结点对都有边相连接）输出 1。该函数显然是单调的，而且已知是 NP-完全的。

定理 B.3（参考文献[187]，在参考文献[7]中进行了改进） 不存在多项式规模的单调电路求解 CLIQUE。

要指出其下界关于结点数量是亚指数的（即对 n 个结点规模为 $\exp(\Omega(n^{1/8}))$ ），且 $\mathcal{P}$ 中的函数有类似的下界。因此，在单调电路复杂性和非单调电路复杂性之间存在一个指数间隔，这个间隔（当然）涉及到单调函数的计算。

B.2.3 有界深度电路

下一个约束是结构性的：允许电路中出现所有的门，但限制电路深度。有向无环图的深度可简化为图中最长有向路径的长度。所以在某种意义上，深度反映了计算该函数的并行时间：如果电路深度为 d ，则该函数可以由足够多的处理器在 d 步内计算（每步中一次计算多个门）。实际上，并行时间是自然且重要的计算资源，涉及到如下基本问题：使用多个并行的计算机能否加速计算？在有多个处理器时，判定哪些计算任务能被“并行化”，而哪些任务“天生就是顺序执行的”，显然是一个基本问题。

将 d 限制为常数，不仅出于度量并行时间的考虑，而且还由于该模型与一阶逻辑的表现力，以及与多项式时间层级（见定义 3.2）之间的关系。在当前环境（常数深度电路）下，允许无界的扇入（即 $\wedge$ 门和 $\vee$ 门可以有任意数量的输入边），否则，每个输出比特将仅依赖于输入常量数量比特。

令 PAR（指奇偶校验）表示两个输入比特的模 2 加，当且仅当输入比特中 1 的数量大于 0 的数量时，令 MAJ（表示大部分）为 1。随机约束方法(Random Restriction Method，由 Furst、Saxe 和 Sipser^[83]提出)的开发带来如下结果。

定理 B.4（参考文献[83]，在参考文献^[115, 240]中进行了改进） 对所有常量 d ，函数 PAR 和 MAJ 没有深度为 d 的多项式规模电路。

① 关键在于，要考虑对每个满足 $\sum_{i=1}^n x_i > \lfloor n/2 \rfloor$ （ $\sum_{i=1}^n x_i < \lfloor n/2 \rfloor$ ）的字符串 $x = x_1 x_2 \cdots x_n$ 计算结果为 1（为 0）的 n 比特单调函数的集合。注意：每个这种函数由 $\binom{n}{\lfloor n/2 \rfloor}$ 个比特描述。

上述改进（在 Yao^[240]之后由 Håstad^[115]提出的）对深度为 d 有 n 个输入的 PAR 电路规划给出了相对紧致的下界 $\exp(\Omega(n^{1/(d-1)}))$ 。

有趣的是，即便电路允许（无界的扇出）PAR 门，MAJ 仍然是困难的（对常数深度多项式规模的电路）（这个结果基于另一种证明技术：多项式近似参考文献[188, 209]）。然而，“反之”则不成立（即带有 MAJ 门的常数深度多项式规模电路能够计算 PAR），通常带有 MAJ 门的常数深度多项式规模电路类（表示为 TC^0 ）似乎很强大。特别是，即使将深度限制到 3，也无法证明 NP 中存在不能由这种电路计算的函数。

B.2.4 公式规模

最后一个约束仍是结构性的——要求有向无环图成为树。直观上，这样可以避免重复计算以前得到的部分结果（并且如果再次需要，则必须再次计算）。于是，得到的布尔电路简化为布尔公式（实际上，又回到了允许取反（ $\neg$ ）和扇入为 2 的 $\wedge$ 、 $\vee$ 门的基本模型）。

使用布尔公式是自然的，这不仅是由于其普遍的数学用途，而且还由于布尔公式的规模可与普通电路的深度以及图灵机所需内存（即其空间复杂性）建立联系。电路复杂性的一个最古老的结果是，PAR 和 MAJ 在这个模型上没有非平凡的下界。通过简单的组合（或信息论的）讨论来证明。

定理 B.5^[144] 用于 n 比特 PAR 和 MAJ 的布尔公式需要的规模是 $\Omega(n^2)$ 。

这应该与用于两种函数的线性规模电路进行对比。^①需要指出， $S(\text{PAR})=O(n)$ 是平凡的，但 $S(\text{MAJ})=O(n)$ 不是。受定理 B.5 影响，人们可能希望对显式函数的公式规模给出超多项式下界。这实际上是一个著名的开放问题。

开放问题 B.6 寻找一个显式布尔函数 f ，使 $S(f)$ 是超多项式的。

为处理该挑战，使用一种有效方法称为通信复杂性方法（Communication Complexity Method，由 Karchmer 和 Wigderson^[137]提出）。该方法断定布尔函数 f 的公式的深度等于如下两方游戏 G_f 的通信复杂性。为第一个参与方给定 $x \in f^{-1}(1) \cap \{0,1\}^n$ ，为第二个参与方给定 $y \in f^{-1}(1) \cap \{0,1\}^n$ ，他们的目标是找到一个 x 和 y 不一致的比特位置（即使得 $x_i \neq y_i$ 的 i ，显然是存在的）。为此，双方要根据事先确定的协议交换消息，问题是这类协议的最佳通信复杂性是什么（依据在最坏输入对的情况下所交换的比特位的数量）。

要注意，证明上述游戏 G_f 通信复杂性的超对数下界，将建立计算 f 的公式规模的超多项式下界（由于在其规模上公式深度可以是对数的）。纯信息论性质的下界（没有在游戏中加到参与方的计算约束）蕴含着计算上的下界。

通信复杂性方法有一个单调版本，其中单调电路的深度与要找使得 $x_i > y_i$ 的 i （而不是使得 $x_i \neq y_i$ 的任意 i ）^② 协议的通信复杂性相关。事实上，单调版本比普通版本更有名，原因是其在建立自然问题单调深度的线性下界方面很有用，如完全匹配（由 Raz 和 Wigderson^[186]提出的）。

B.3 算术电路

现在去掉布尔数学的外壳来讨论普通域上的电路。固定任意域 F ，有向无环图的门为域

① 我们指出， $S(\text{PAR})=O(n)$ 是平凡的，但 $S(\text{MAJ})=O(n)$ 不是。

② 要注意，因为 f 是单调的， $f(x)=1$ 和 $f(y)=0$ 意味着存在使得 $x_i=1$ 和 $y_i=0$ 的 i 。

中的标准+和 $\times$ 。图的输入为域 F 中元素,由对其赋值诱导出其他结点的值(在 F 中)。因此,有 n 个输入 m 个输出的算术电路可以计算一个多项式映射 $p: F^n \rightarrow F^m$,且任意一个多项式映射都可由某个电路计算。(按照习惯可以设置某些输入为常数,最重要的是常数-1)。①

算术电路提供了一种自然方法来计算多项式映射,而其规模就成为映射复杂度的一种自然度量。用 $S_F(p)$ 表示计算 p 的最小电路和规模(如果没有下标,则 $F=Q$ (有理数域))。照例我们主要讨论多项式序列,每个多项式对应一个输入规模,并渐近地研究相应的电路规模。

显然,在任何固定的有限域上,算法电路能够模拟布尔电路,在规模上仅相差一个常数因子。因此,主要考虑无限域上的电路,其下界将更容易得到。

与布尔代数情况相同,困难函数的存在性易于证明(通过对维数的讨论,而不是凭借计数来论证),我们主要关心显式多项式(族)。非常粗略地讲,如果存在高效算法,将给定次数序列(定义了一个单项式)映射为相应的系数(的有限描述),则称多项式是显式的。

多项式的次数(Degree)是一个重要的参数,这在布尔模型中不存在。显然,阶为 d 的多项式(即使在只有一个变量的情况下,即 $n=1$)需要的规模至少为 $\log d$ 。简单考虑单变量的情况(其中 d 是输入规模唯一的量度),其中已包含了明确且重要的问题。然后再转向普通多变量的情形,其中输入的数量 n 是一个主要参数(假定 $d \leq n$)。进一步的细节可见参考文献[86, 215]。

B.3.1 单变量多项式

计算 d 次多项式的算术电路,其规模的下界 $\log d$ 究竟紧致到什么程度?简单的维数讨论表明,对绝大多数 d 次多项式 p ,有 $S(p)=\Omega(d)$ 。然而,已经有一个非显式的结果。

开放问题 B.7 寻找 d 次显式多项式 p ,使得 $S(p)$ 不为 $O(\log d)$ 。

为解决这一公开问题,考虑如下两个具体多项式 $p_d(x)=x^d$ 和 $q_d(x)=(x+1)(x+2)\cdots(x+d)$ 。显然, $S(p_d) \leq 2\log d$ (通过反复开方),所以平凡的下界本质上是紧致的。另一方面,确定 $S(q_d)$ 是一个主要的开放问题,推测 $S(q_d)$ 不是 $\log d$ 的多项式。为认识到该问题的重要性,给出如下命题。

命题 B.8 如果 $S(q_d)=\text{poly}(\log d)$,则整数因子分解问题可以利用多项式规模电路求解。

普遍认为,整数因子分解问题是困难的(特别是不存在多项式规模的电路)。为证明命题B.8,观察到 $q_d(t) \equiv ((t+d)!)/(t!) \pmod{N}$,且通过一个用于 q_d 的电路能够有效得到 $((t+d)!)/(t!) \pmod{N}$ 的值(通过模拟对前一个电路模 N 的计算)。注意到, $(\sum_{i=1}^l K_i)! = \prod_{i=1}^l q_{K_i}(\sum_{j=i+1}^l K_j)$,从而 $(K!) \pmod{N}$ 的值可利用计算多项式 $\langle q_{2^i} : i=1, 2, \dots, \lfloor \log_2 K \rfloor \rangle$ 的电路求得。下一步,观察到当且仅当 N 的所有素数因子均比 K 大时, $(K!) \pmod{N}$ 和 N 是互素的。于是,给定一个合数 N ,通过对适当的 K 进行折半查找,就能找到 N 的一个因子。□

B.3.2 多变量多项式

再回到有 n 个变量的多项式上来。为使 n 成为仅有的“问题规模”参数,可以限制多项式的次数不超过 n 。

几乎所有含 n 个变量的多项式均要求规模 $S(p) \geq \exp(\Omega(n))$,我们寻找困难的显式多项

① 这样可以模仿常数加、常数乘及减运算。我们要指出,在计算(无限域上的)多项式时,除法可用其他操作来高效地模仿。

式(族)。与布尔函数不同,这里的是粗略的非平凡下界(通过代数几何中的基本工具)。

定理 B.9^[26] $S(x_1^n + x_2^n + \cdots + x_n^n) = \Omega(n \log n)$ 。

同样的技术也可用于证明其他自然多项式的下界,如对称多项式和行列式。对任意显式多项式建立较强的下界是一个重要的开放问题。另一个开放问题是对总次数为常量(即使是1)的多项式映射求超线性的下界。后一个开放问题比较著名的结果是计算复数域上的离散傅利叶变换的线性映射,或者有理数上的 Walsh 变换(这两个问题已知有 $O(n \log n)$ 的算法,但超线性下界尚且未知)。

现在讨论非常重要的一类特定多项式。对后一个开放问题最自然且研究成果最多的是矩阵相乘函数 MM: 令 A, B 为 F 上两个 $m \times m$ 的矩阵,定义 $MM(A, B)$ 为矩阵 $A \times B$ 的 $n=m^2$ 个值。于是,MM 为 $2n$ 个输入变量上 n 个显式双线性形的一个集合。已知 $S_{GF(2)}(MM) \geq 3n$ (见参考文献[206])。另一方面,含 $O(m^3) = O(n^{3/2})$ 步的算法可以改进。

定理 B.10^[62] 对每个域 F , $S_F(MM) = o(n^{1.19})$ 。

然而,MM 的复杂度是什么(即使仅统计乘法门)?是线性还是几乎线性,抑或对某些 $\alpha > 1$ 有 $S(MM) > n^\alpha$ 吗?这是一个著名的开放问题。

接下来考虑由 $n=m^2$ 个变量表示的 $m \times m$ 矩阵的行列式和常值多项式(分别记作 DET 和 PER)。DET 在经典数学中扮演着重要角色,而 PER 却有点难懂(尽管它出现在统计力学和量子力学中)。在复杂性理论中这两种多项式都十分重要,因为它们反映了自然的复杂性类。函数 DET 的复杂度相对较低(接近于拥有多项式规模算术公式的多项式类),而 PER 似乎复杂度更高(即对所有“p-定义”的多项式类是完全的(见参考文献[231],且对于计数类 #P 是完全的(见 6.2.1 节))。于是,推测 PER 不能在多项式时间内归约为 DET。在其代数环境中,一种起作用的受约束归约方法是通过投影归约。

定义 B.11 (投影) 令 $p_n: F^n \rightarrow F^l$ 和 $q_N: F^N \rightarrow F^l$ 为多项式映射, $x_1, x_2, \dots, x_n$ 为 F 上的变量。如果存在一个函数 $\pi: [N] \rightarrow \{x_1, x_2, \dots, x_n\} \cup F$, 使得 $p(x_1, x_2, \dots, x_n) \equiv q(\pi(1), \pi(2), \dots, \pi(N))$, 称在 F 上存在一个从 p 到 q 的投影。

显然,如果 $p_n \leq q_N$, 则 $S_F(p_n) \leq S_F(q_N)$ 。令 DET_m 和 PER_m 表示限制到 m 乘 m 矩阵的函数 DET 和 PER。已知 $PER_m \leq DET_{3m}$, 但为产生一个多项式时间的归约, 需要从 PER_m 到 $DET_{poly(m)}$ 的投影。毫无疑问,人们猜测不存在这样的投影。

B.4 证明的复杂性

证明的概念将对数学研究从人类的其他研究领域分离出来。数学家们在刻画诸如“深刻的、原始的、深入的”尤其是“困难的”这些形容词方面积累了大量经验。能否通过数学方法量化证明不同定理的困难性?这是证明复杂性要解决的问题。力求依据证明的困难性来区分这些定理,就像电路复杂性力求根据计算的困难性来区分函数一样。在证明中,与在计算中相同,有大量称为证明系统的模型,这些模型界定了证明者所允许的合理能力。

我们仅考虑命题证明系统,于是,(这些系统中的)定理就成为重言式。这些系统都是完全且合理的,即有且仅有一个重言式的证明与这些系统相关,且必存在一个重言式的证明与

之相关。证明系统的形式化定义阐明了一些被认为是理所当然的事情：验证程序的效率。在如下定义中，验证程序的效率是指用定理及其证明的总长度来度量的运行时间。^①

定义 B.12^[61] (命题) 证明系统是一个多项式时间图灵机 M ，公式 T 是一个重言式当且仅当存在一个称为证明的串 π ，使得 $M(\pi, T)=1$ 。

与标准形式化体系（见后面）一致，证明被视为在定理之前就已出现。注意：定义 B.12 确保了证明系统的完整性与合理性，以及验证效率（与证明一定理对的总长度相关）。定义 B.12 忽略了（对重言式 T 的）证明 π 的长度，将证明的长度视为关于证明系统 M 的重言式 T 的复杂性的度量。

对每个重言式 T ，令 $\mathcal{L}_M(T)$ 表示 M 中 T 的最短证明的长度（即使得 M 接受 (π, T) 最短字符串 π 的长度）。也就是说， $\mathcal{L}_M$ 反映了关于证明系统 M 的不同重言式的证明复杂度。再滥用一下符号，令 $\mathcal{L}_M(n)$ 表示在长度为 n 的重言式上 $\mathcal{L}_M(T)$ 的最大值。下列定理给出了证明复杂性（关于任意命题的证明系统）和计算复杂性（即 NP-vs-coNP 问题）之间的基本关系。

定理 B.13^[61] 存在命题证明系统 M ，使得函数 $\mathcal{L}_M$ 为多项式的当且仅当 $\mathcal{NP}=\mathcal{coNP}$ 。

特别地，使得 $\mathcal{L}_M$ 为多项式的命题证明系统 M ，与对命题重言式集合这一 $\mathcal{coNP}$ -完全集合的 NP 证明系统是一致的。

研究证明，复杂性的长远目标是对任意命题证明系统的证明长度建立超多项式的下界（并由此证明 $\mathcal{NP}\neq\mathcal{coNP}$ ）。在实现这个庞大工程时，首先考虑简单（并且较弱）的证明系统，然后转向越来越复杂的系统是很自然的。不同证明系统中规定了基本的（受限的）类型、论证的“原语”以及自然的重言式，并将这些作为研究的主题。本节剩余部分主要关注受约束的证明系统。

不同数学分支，如逻辑、代数和几何，通常隐式地提供了不同的证明系统。典型的证明系统包含一个公理集和一个推理规则集。证明将通过一系列步骤推导出要证明的重言式，每一步产生一个公式（通常称为证明的一行），该公式要么是公理，要么由前面的公式通过一条推理规则得到。关于这些证明系统有两点观察。第一，这些系统中的证明易由一个算法来验证（于是符合定义 B.12 的通用框架）。第二，这些证明系统完全符合有向无环图模型，其输入标记为公理，内部结点标记为推理规则（如 B.1 节）：应用内部结点的推理规则，可以证明每个输出代表的重言式。^②

对不同的证明系统 Π ，要研究其中重言式 T 的证明长度 $L_\Pi(T)$ 。第一个观察揭示了证明复杂性与电路复杂性的主要区别，即平凡的计数方法是无效的。原因在于，关于 n 个比特的函数个数为 2^{2^n} ，而等长的重言式最多只有 2^n 个。于是，在证明复杂性中，困难的重言式即使存在，也不一定是显式的，我们对此毫无兴趣（而且，特别是，如果该重言式在所有命题证明系统中均成立，将会得到 $\mathcal{NP}=\mathcal{coNP}$ ）（注意到，这里涉及到的是问题的困难实例，而不是困难问题）。无论如何，大部分已知下界（在受限的证明系统中）都可用于非常自然的（更

① 事实上，这个习惯与第 9 章使用的标准习惯不同，在第 9 章中，验证的复杂度（即验证者的运行）被表示为定理长度的函数。两种方法是一致的，且在 2.1 中均有提及，因为在 2.1 节，强制性地令证明长度为定理长度的多项式。

② 考虑机器 M 的单步推理规则，定义 B.12 中的普通证明系统也能用于这种形式化体系。然而，这里的推理规则更简单，更重要的是，它们是自然的。

不用说是显式的)重言式。

重要约定

证明系统有一个等价的但更方便的方法,称为(简单)反驳系统。首先,回顾 3SAT 是 NP-完全的,每个(否定)重言式可以写成子句的合取,每个子句又是 3 个文字(变量或其取非)的析取。如果将这些子句作为公理,(使用系统规则)得到一个明显的矛盾(如对公理的否定,或更好的是空子句),那么就证明了重言式(因为证明其否定会导致矛盾)。证明复杂度常采用反证法,并常常与其否定(“矛盾”)交换“重言式”。

内容组织

本节剩余内容分 3 个部分,分别涉及逻辑、代数和几何证明系统。我们简要描述各个领域的代表性成果,进一步的细节可参阅参考文献[27](这是一个特别详细的参考文献)。

B.4.1 逻辑证明系统

本节的证明系统都有布尔公式行,区别是给这些公式施加了结构性的限制。

最基本的证明系统,称为 Frege 系统,其中对要证明的公式没有施加约束条件。它有一个派生规则,称为截规则(Cut Rule): $A \vee C, B \vee \neg C \vdash A \vee B$ (其中 A 、 B 和 C 为命题公式)。在该系统上增加任意合理的规则,如演绎推理,对于证明长度几乎没有影响。

Frege 系统是一种基本系统,它们(的几种变形)在逻辑学中很常见。事实上, Frege 系统中多项式长度的证明自然地对应着关于可行对象的“多项式时间推理”。证明复杂性中一个重要的开放问题是寻找在 Frege 系统中没有多项式长度证明的重言式(重言式族)。

由于求 Frege 系统的下界是困难的,我们转向更有意思且更自然的 Frege 子系统。得到最广泛研究的是 Resolution,可应用于绝大多数命题(以及一阶)自动定理证明系统中。在 Resolution 中允许的公式都是简单的子句,所以对于子句 A 、 B 和变量 x ,派生截规则简化为“Resolution 规则”: $A \vee x, B \vee \neg x \vdash A \vee B$ 。

一般的 Frege 系统与 Resolution 系统的功能有差异,这体现在存在一些重言式,对于前者是容易的,而对于后者则是困难的。一个例子是鸽巢原理 PHP_n^m ,表示不存在 m 只鸽子到 $n < m$ 个鸽巢的一一映射。

定理 B.14 $\mathcal{L}_{\text{Frege}}(\text{PHP}_n^{n+1}) = n^{O(1)}$, 而 $\mathcal{L}_{\text{Resolution}}(\text{PHP}_n^{n+1}) = 2^{\Omega(n)}$ 。

B.4.2 代数证明系统

正如一个子句的不可满足集构成布尔系统中的矛盾,在代数系统中一个自然矛盾是没有公共根的多项式系统。另外,在任意域上, CNF 公式易于转化为多项式系统,每个多项式对应一个子句。通常会加上确保布尔值的公式 $x_i^2 - x_i$ 。

一个自然的证明系统(与希尔伯特零点定理(Hilbert's Nullstellensatz)相关,在符号代数系统中用来计算 Grobner 基)是多项式子句(Polynomial Calculus),简称为 PC。该系统中每一行都是多项式(由所有系数显式的表示),有两个推理规则:对任意两个多项式 g 和 h ,规则为 $h \vdash g + h$,对任意多项式 g 和变量 x_i ,规则为 $x_i \vdash x_i g$ 。该系统长度的强的下界(由阶的下界可以得到)是已知的。例如,将鸽巢原理作为常数阶多项式的一个矛盾集,就得到如下定理。

定理 B.15 对所有 n 和所有 $m > n$, 在所有域上有 $\mathcal{L}_{\text{PC}}(\text{PHP}_n^m) \geq 2^{n/2}$ 。

B.4.3 几何证明系统

然而, 另一种表示矛盾的自然方法是交集为空的空间集合。主要关注离散的 (即布尔型的) 定义域, 广泛采用来自组合优化中的整数程序作为矛盾。其中的约束条件中整系数 (仿射) 线性不等式组 (所以值域是布尔立方体被半空间切割的子集)。最基本的系统称为切削平面 (Cutting Planes, CP)。其行是整系数线性不等式组。推理规则是 (明显的) 线性不等式组加法, 以及 (不太明显的) 系数与一个常量的除法。

当 PHP_n^m 在该系统中为容易时, 其他重言式的指数下界是已知的。这可由 B.2.2 节中的单调电路下界得到。

附录 C

现代密码学基础

没有地基，可以建起一间小屋，但决不可能建起持久的大厦。

Eng. Isidor Goldreich (1906—1996)

内容提要

密码学研究如何设计能够抵抗任何滥用行为的计算系统：这样的系统即使面对着使其背离功能的恶意企图，也仍能保持既定功能。

本附录介绍密码学基础理论，主要是指为安全问题提供定义及解决方案的范式、方法和技术。其中包括一些概念工具以及利用它们而得到的基本结论。本附录的重点是澄清基本概念，并阐明解决密码学中几个核心问题的可行性。假设读者具备算法、概率论与复杂性理论的基本知识。

本附录进一步详细讨论了分别出现于 7.1 节、8.2 节和 9.2 节中的单向函数、伪随机数发生器和零知识证明。^①应用这些基本工具，还讨论了基本的密码应用系统，如加密、签名和密码协议。

C.1 引言与预备知识

理论计算机科学的主要成果之一是对密码学的广泛扩展与严格处理。特别是将经典概念如安全加密和不可伪造签名等建立在坚实的基础上，还揭示了一些新的（未知）方向及联系。进一步地，密码学的发展还伴随着新概念的引入，如计算不可区分性、伪随机性和零知识交互证明等，这些概念都具有独立意义（分别见 7.1 节、8.2 节及 9.3 节）。事实上，现代密码学和复杂性理论之间有较强的联系（相比之下，“经典”密码学与信息论的相关性更强）。

C.1.1 基本原则

现代密码学关心如何构造健壮的信息系统，能够抵抗那些试图使其偏离既定功能的行为。既定功能可以是在不安全信道上的秘密通信及信息认证，公平且秘密的电子投票，或能“抗失效”的多方计算。事实上，现代密码学范围是很广的，这和经典密码学形成了鲜明对比（经典密码学仅仅关注在不安全信道上的保密通信问题）。

^① 这些讨论对密码学很重要，但不是本书的主要内容，因此在本书正文中省略了。

C.1.1.1 敌手

设计密码系统是一个非常困难的任务。设计者不能依赖系统运行环境中的“典型”状态，确切地说，攻击系统的敌手会尽力将环境变成“非典型”状态。人们也不能满足于拥有能对抗那些“非典型”状态的措施，因为敌手（通常在设计结束后开始行动）将尝试使用不同于设计者预想的方式实施攻击。这是显而易见的，但仍有人期望在实践中这样做不会带来实际上的损害。经验表明这种期望往往会落空，基于假想而设计的密码系统迟早会被破坏的。

鉴于上述讨论，我们认为，假设对手使用特定策略意义不大。唯一合理的假设是敌手的计算能力。密码系统的设计必须建立在坚实的基础之上，采用随意和试探性的方法非常危险。

密码学的基础就是用于为自然的“安全问题”提供定义和解决方案的范式、方法和技术。密码学问题（或安全问题）的解决包含两个阶段，定义阶段和构造阶段。首先，在定义阶段，要描述所需功能，并定义充分的密码学问题。尝试枚举所有非预期情况是不可行的，也易出错。因此，需要在一个假定的理想模型下用操作的术语定义功能，并要求备选的解决方案能在现实的、清晰定义的模型下（其中规定了敌手的能力）模仿这些操作。定义完成之后，就要构造一个满足定义的系统。构造中可能使用一些简单工具，而系统的安全性利用这些工具来证明。

实例

为实现在不安全信道上的保密（或可靠）通信，在定义阶段，需要对安全加密方案（或签名方案）的概念进行形式化。然后，利用简单的原语，如单向函数，构造这种方案，而构造的安全性通过“可约性方法”来证明（表明如何将单向函数的求逆“归约”为破坏构造的安全性，见 7.1.2 节）。

C.1.1.2 计算假设的应用

与上述例子相同，只有当存在某些计算困难问题时，才存在密码学中使用的工具和应用程序（如安全加密）。确切地说，这些工具和应用程序（显式或隐含地）要求生成困难问题实例的能力。单向函数的定义中体现了这种能力。从而，单向函数是完成密码学任务的最小需求（结果如我们将看到的，这个必要条件在本质也是“充分的”，也就是说，单向函数的存在性（或对这一假设的扩展）对于大部分密码学应用而言已经足够了）。

以我们当前对高效计算的理解，还无法证明单向函数的存在性。特别是，如 7.1.1 节和附录 C.2 中所讨论的，证明单向函数的存在性似乎比证明 $P \neq NP$ 还要难。因此，别无选择（在这个历史阶段），只能假设单向函数存在。并只能以数百（或数千）研究者的信念作为这个假设成立的理由。此外，这些信念涉及一个简单的假设，其合理性来自一些在各个不同领域中被普遍接受的重要猜想（如对整数因子分解困难性的猜想是计算数论的核心）。

鉴于无论如何都需要假设，“为什么不仅仅针对需求做出假设呢（即只假设存在针对一些密码问题的解决方案）？”那么，首先必须知道需求是什么，也就是说，必须澄清我们需要什么，这意味着要在定义阶段深思熟虑。但是一旦该阶段结束并得到了定义，则能否假设存在满足该定义的系统呢？非也：定义的存在性并不蕴含着存在符合定义的系统。

阐述一个密码学定义切实可行（且可解决相应安全问题）的方法是基于一个易于理解（即被普遍接受）的假设来证明该定义可被满足。举个例子，考虑零知识证明的定义，尚不清楚是否存在这种证明（在非平凡的意义）。为表明这一概念的非平凡性，可以先给出一个关于二剩余断言的零知识证明系统，而后者被认为是难以验证的（无附加信息）。进一步地，与人

们之前的观念相反,后来证明了单向函数的存在性蕴含着任意 NP-断言都能在零知识下得到证明。于是,一些被认为根本不成立(甚至认为是错误)的事实可以通过将其“归约”到被普遍接受的假设而得到证明(不会引起密码学的崩溃)。

综上所述,并非所有的假设都是等价的。因此,将一个复杂的、新的和可疑的假设“归约”为一个被普遍接受的简单假设是极有价值的。进一步地,将一个对新任务的解决方案“归约”到众所周知的原语的安全性,意味着提供了一种用已知原始解决新问题的典型方法。这表明,我们不仅可以确信新任务是能够被解决的,而且利用众所周知的多用途原语构造了一种解决方案。

C.1.2 计算模型

密码学主要考虑构造高效的密码方案,且破坏其安全性是不可行的。因此,我们需要明确高效计算和不可行的定义。方案中合法用户的计算应该是高效的,而(敌手)破坏安全性应该是不可行的。强调一下,这里不区分可行的计算与高效的计算,而将前者视为一个更自由的概念。以下详述。

C.1.2.1 高效的计算与不可行的计算

高效的计算通常是指所需时间为安全参数多项式的计算。界定合法用户运行时间的多项式是固定的,且是明确的(通常很小)。事实上,我们的目标是得到一个尽可能严格的效率的概念(或者说,尽可能高效的开发计算策略)。这里(即涉及合法用户的复杂性时)与任何算法的运行环境没有什么不同。但是,当涉及到敌手的计算资源假设时,应有所区别,我们希望可行的概念尽可能宽泛。一种通用方法是假设可行的计算也是多项式时间的,但这里的多项式并未预先规定(被认为是任意大)。换句话说,只限定敌手运行于多项式时间,而在此之外的任何运算都认为是不可行的。

尽管已有很多精确的定义将可行的计算与多项式时间相关联,但这种习惯并不重要。关于可行性有一个更一般的概念:可行性在算法合成下应保持,从而,敌手的运行时间上限为任意的超多项式函数(或函数类)。为了更加具体和清晰,在形式化定义中我们采用前面的习惯做法(但动机讨论将涉及未明确定义的可行性概念,其中至少包含了高效的计算)。

C.1.2.2 随机化的(或概率的)计算

随机化的计算是密码学的核心。根本原因在于随机性是秘密存在(或生成)的本质。于是,必须允许合法用户使用随机化的计算,并且(由于认为随机化是可行的)必须认为敌手也使用随机化的计算。这就引出成功概率的问题:一般要求合法用户的成功概率(在实现合法目标时)为1(或者接近1),而敌手的成功概率(在破坏安全属性时)可忽略。所以,可忽略概率的概念在这里起着重要的作用。

为了定义可忽略的概率,给出健壮的小概率概念:即使是尽可能多地重复实验,小概率事件也很少发生。也就是说,如果认为多项式时间计算是可行的,则对任意多项式 p 和足够大的 n , 当 $1-(1-\mu(n))^p < 0.01$ 时,称函数 $\mu: \mathbf{N} \rightarrow \mathbf{N}$ 为可忽略的(即如果对任意正多项式 p' , 函数 $\mu(\cdot)$ 的上界为 $1/p'(\cdot)$, 则 μ 是可忽略的)。

还需讨论显著性概率的概念。如果重复多项式次数的实验,事件将几乎确定会发生(即除了可忽略的概率),则称事件以显著概率发生。因此,如果对所有的正多项式 p' , 函数 $\nu(\cdot)$ 的下界为 $1/p'(\cdot)$, 则称函数 $\nu: \mathbf{N} \rightarrow \mathbf{N}$ 为显著的。

C.1.3 内容组织

本附录主要讨论密码学中的典型问题（如加密和签名方案），以及一些重要工具（如计算困难性、伪随机性及零知识证明）。在阐述每个问题时，我们先介绍该问题产生的背景，再给出定义，最后阐明如何解决该问题。这里的重点是要说明解决问题的可行性，而不是提供一种可行的解决方案。

我们的目的是介绍密码学中的基本概念、技术和结果，重点是澄清重要概念及相互关系。这需要采用独立于数论细节的方式。数论中的具体例子在密码学发展过程中起重要作用，而且仍用于实现大部分的密码学原语，但这并不意味着在表述时必须涉及这些例子。相反，我们认为概念的澄清必须脱离具体实现，在抽象层面上进行。

内容组织

本附录包括两个主要部分，分别是密码学的基本工具和密码学的基本应用。

基本工具

最基础的工具是计算困难性，可用单向函数的概念解释。另一个关键的概念是计算不可区分性，它是伪随机性和其他密码学概念的基础。伪随机数发生器和函数是经常使用的重要工具。零知识证明也一样，它在安全密码协议的设计和研究中起重要作用。

基本应用

加密和签名方案是最基本的密码学应用。其主要作用是在不安全的通信信道上提供安全且可靠的通信。粗略地讲，加密方案用于确保通信中信息的机密性（或隐私性），而签名方案是要确保它的可靠性（或可认证性）。另一个基本问题是构造能实现任意功能的安全密码协议。

对基本工具的介绍出现在附录 C.2～附录 C.4（有些部分会有重复）、7.1 节、8.2 节和 9.2 节（分别介绍了单向函数、伪随机数发生器和零知识证明）。C.5 节～C.7 节概述了基本密码学应用，即加密方案、签名方案及通用密码协议。

对进一步阅读的建议

本附录是作者关于该主题的两卷本著作^[91, 92]的简明总结。而且，第一部分（即基本工具）对应着参考文献[91]，而第二部分（即基本应用）对应参考文献[92]。感兴趣的读者可以从这些教科书中得到进一步的细节（特别是所缺少的参考文献）。

实践

本附录的目的是介绍密码学的理论基础。如上所述，这些基础在密码学的学习和实践中是非常必要的。事实上，实践中需要的远不止基础理论，而本附录内容仅限于基础理论。然而，有了良好的基础，就能学会评估实际方案。另一方面，缺乏良好的基础则无法精确地评估实际方案，从而导致错误的决策。在设计要抵抗敌手攻击的方案时，最具危害性的便是对攻击的错误理解。

C.2 计算困难性

现代密码学研究构造易于（正确地）操作而难于攻破的系统。于是，一个复杂性缝隙（在正确使用的容易性和破坏预设功能的困难性之间）便存在于现代密码学的核心问题中。然而，并不知道所需的这种缝隙是否存在，而仅是普遍认为其存在。事实上，几乎所有现代密码学

分支的兴起或衰落都伴随着单向函数的存在与否。单向函数的存在性蕴含着 $\mathcal{NP}$ 中包含了难解的搜索问题，反过来又蕴含了 $\mathcal{NP}$ 不包含于 $\mathcal{BPP}$ 中（即最坏情况下的复杂性猜想）。

粗略地讲，单向函数是易于计算而难于（一般而言）求逆的函数。这些函数可以作为产生不可解“谜题”的有效方式（即谜题是函数的随机映像，而其解是相应的原像）。而且，生成谜题的人知道它的解并且可以高效验证（可能的）解的正确性。于是，由定义可知单向函数具有明显的密码学特色（即在一种任务的易解性与另一任务的困难性之间出现缝隙）。

C.2.1 单向函数

首先重新给出 7.1.1 节中单向函数的基本定义，再进一步讨论该定义。

定义 C.1 (单向函数, 重述定义 7.1) 如果以下两个条件得到满足, 则函数 $f: \{0,1\}^* \rightarrow \{0,1\}^*$ 被称为单向函数。

(1) 易于计算: 对所有 $x \in \{0,1\}^*$ 存在多项式时间算法 A , 使得 $A(x)=f(x)$ 。

(2) 难于求逆: 对所有概率多项式时间的算法 A' , 所有的多项式 p 和所有足够大的 n , 有

$$\Pr[A'(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

此处概率在 $x \in \{0,1\}^*$ 和算法 A' 的所有内部随机数上均匀产生。

一些最常用的单向函数基于数论中计算问题的难解性假设。其中一个假设是分解大整数不可行。因此, 输入两个 (等长的) 素数并输出其乘积, 这是一个单向函数。而且, 当且仅当模这样一个合数求平方为单向函数时, 分解该合数不可行 (见参考文献[183])。对于特定的合数 (即两个均等于 $3 \bmod 4$ 的素数之积), 后一个函数诱导出该合数的二次剩余集合上的置换。RSA 函数: $x \mapsto x^e \bmod N$ [193] 是另一个被认为是单向函数的相关置换, 其中 $N=PQ$ 是满足上述条件的合数, e 与 $(P-1)(Q-1)$ 互素, 且 $x \in \{0,1,\dots,N-1\}$ 。单向函数集的如下定义 (实际上与定义 C.1 相关) 更好地描述了后一个例子 (以及其他常用方案)。

定义 C.2(单向函数集) 如果存在 3 个概率多项式算法 I 、 D 和 F 满足如下两个条件, 则称函数集 $\{f_i: D_i \rightarrow \{0,1\}^*\}_{i \in \bar{I}}$ 为单向的。

(1) 易于抽样和计算: 对输入 1^n , 算法 I 的输出分布在集合 $\bar{I} \cap \{0,1\}^n$ 上 (即是一些 n -比特长函数的索引)。对输入 (函数索引) $i \in \bar{I}$, 算法 D 的输出 (定义域抽样) 分布在集合 D_i 上 (即函数的定义域)。对于输入 $i \in \bar{I}$ 和 $x \in D_i$, (计算) 算法 F 总是输出 $f_i(x)$ 。

(2) 难于求逆^①: 对所有概率多项式时间算法 A' , 所有的正多项式 $p(\cdot)$, 和所有足够大的 n , 有

$$\Pr[A'(i, f_i(x), 1^n) \in f_i^{-1}(f_i(x))] < \frac{1}{p(n)}$$

式中: $i \leftarrow I(1^n)$, $x \leftarrow D(i)$ 。

如果每个 f_i 是相应的 D_i 上的置换, 并且 D_i 在 $D(i)$ 中几乎均匀分布, 则称该函数集为置换集。

例如, 对于 RSA, 考虑 $f_{N,e}: D_{N,e} \rightarrow D_{N,e}$ 满足 $f_{N,e}(x) = x^e \bmod N$, 其中 $D_{N,e} = \{0,1,\dots,N-1\}$ 。

^① 注意: 这个条件涉及到分布 $I(1^n)$ 和 $D(i)$, 其中仅要求分布的区域分别为 $\bar{I} \cap \{0,1\}^n$ 和 D_i (典型地, 分布在 $\bar{I} \cap \{0,1\}^n$ 和 D_i 上是 (几乎) 均匀的)。

由定义 C.2 出发,可以得到陷门置换^①的定义。粗略地讲,后者是一组单向置换,当给定适当的辅助信息(称为陷门)时,存在求逆置换的高效算法。

定义 C.3(陷门置换) 对于定义 C.2 中的一组置换,如果存在两个附加的概率多项式算法 I' 和 F^{-1} , 满足如下两个条件,则称其为陷门置换: 分布 $I'(1^n)$ 在字符串对上取值,使得第一个字符串与在 $I(1^n)$ 中的分布相同; 对在 $I'(1^n)$ 的取值范围内的所有 (i, t) 和所有 $x \in D_i$, 存在 $F^{-1}(t, f_i(x)) = x_i$ (即 t 是允许对 f_i 求逆的陷门)。

例如,对于 RSA, $f_{N,e}$ 可通过指数 d 来求逆(模 $N=PQ$), 其中 d 是 e 模 $(P-1)(Q-1)$ 的乘法逆元。实际上,此时陷门信息为 (N, d) 。

强单向函数与弱单向函数(7.1.2 节小结)

回顾上述定义,要求任意可行的算法求逆成功的概率可以忽略。较弱的概念只要求任意可行的算法不能以显著的概率求逆。从而弱单向函数的存在性蕴含了强单向函数(如定义 C.1)的存在性。构造本身相当直接,但对其分析却超出了信息论范畴。相反,所构造方案的安全性(即求逆的困难性)可以通过可约性方法来证明,这种方法将对某个结论(即所构造方案的安全性)的破坏转换为对某种假设(给定的原语的安全性)的破坏。可以利用这种策略(即“可约性方法”)证明该领域中所有的条件性结果。

C.2.2 困难核心谓词

回顾函数 f 的单向性意味着(在 f 的值域内)给定 y , 要找到 y 在 f 下的原像是不可行的。但这并不是说要找到关于 y 在 f 下的原像的部分信息是不可行的。确切地说,有可能容易地找到原像的一半的信息(即给定单向函数 f , 对所有 $|x|=|r|$ 定义函数 g 为 $g(x, r) \stackrel{\text{def}}{=} (f(x), r)$)。后面将说明,隐藏(关于函数原像的)部分信息在很多高级的构造(如安全加密)中起着重要作用。这些部分信息称为对 f 求逆的困难核心。粗略在讲,多项式时间可计算的(布尔)断言 b 称为函数 f 的困难核心,如果不存在可行算法,对给出的 $f(x)$, 能够以超过的一半的不可忽略的概率猜出 $b(x)$ 。准确的定义见 7.1.3 节(即定义 7.6)。

注意: 如果 b 是一个多项式时间可计算的 1-1 函数 f 的困难核心,则 f 是一个单向函数。另一方面,回顾定理 7.7, 对于任何单向函数 f , x 和 r 模 2 的内积是 $f'(x, r) = (f(x), r)$ 的核心。

C.3 伪随机性

实践中常用“伪随机”序列来替代真随机序列。一种基本观念认为,如果一个(高效的)应用程序在使用真随机序列时运行良好,那么,在使用“伪随机”序列时也会运行得一样好。然而,这个观念并没有得到关于“伪随机性”一些特定概念的支持,如通过参考文献[146]中的统计测试或拥有大的“线性复杂度”(定义见参考文献[112])。当然,在密码学应用中使用这样的“伪随机”序列(而不是真随机序列)是很危险的。

相比之下,如果伪随机序列由与均匀分布计算上不可区分的伪随机分布提供,则可以用伪随机序列安全地替代真随机序列。从对伪随机分布的这种要求易推出替换的合理性。简单地说,伪随机数发生器被定义为基于一些真随机比特(即短的随机种子)生成伪随机序

^① 实际上,更恰当的术语是陷门置换集,但简短的(并且不精确的)术语更常用。

列的高效率程序。这种构造在以下意义上与密码学相关：它可以为合法用户提供共享的短随机种子，以生成对任何敌手（不知道种子）而言是随机的长序列。

C.3.1 计算不可区分性

现代密码学的一个核心概念是“高效的相似性”（亦称计算不可区分性，参见参考文献[108, 239]）。这里我们并不关心对象是否等价，而是关心通过一个可行的计算来观察目标是否不同。负面的回答是，就任何实际应用而言两个目标是等价的。实际上，后面常常将这样的两个（计算不可区分的）目标互换。本节回顾计算不可区分的定义（见 8.2.3 节），并考虑两种变形。

定义 C.4 (计算不可区分性，定义 8.4 的修改) ^①称 $X = \{X_n\}_{n \in \mathbb{N}}$ 和 $Y = \{Y_n\}_{n \in \mathbb{N}}$ 是计算不可区分的，如果对所有概率多项式时间算法 D ，所有多项 p ，和所有足够大的 n ，有

$$|\Pr[D(1^n, X_n) = 1] - \Pr[D(1^n, Y_n) = 1]| < \frac{1}{p(n)}$$

此处概率由相关的分布（即 X_n 或 Y_n ）和算法 D 的内部随机数产生。

进一步讨论见 8.2.3 节。特别是，回顾“可高效构造的”分布，其中单个样本（见以上定义）的不可区分性隐含了多个样本的不可区分性（如定义 8.5）。

扩展到由字符串索引的总体

讨论定义 C.4 的一个自然扩展：样本空间由自然数索引扩展为任意集合 $S \subseteq \{0,1\}^*$ 上的索引。典型地，对于空间 $\{Z_\alpha\}_{\alpha \in S}$ ， Z_α 的取值范围为串的长度，与 α 的长度多项式相关。

定义 C.5 称 $\{X_\alpha\}_{\alpha \in S}$ 和 $\{Y_\alpha\}_{\alpha \in S}$ 是计算不可区分的，如果对所有概率多项式时间算法 D ，所有多项式 p 和所有足够长的 $\alpha \in S$ ，有

$$|\Pr[D(\alpha, X_\alpha) = 1] - \Pr[D(\alpha, Y_\alpha) = 1]| < \frac{1}{p(\alpha)}$$

此处概率由相关的分布（即 X_α 或 Y_α ）和算法 D 内部随机数产生。

注意：定义 C.4 为 $S = \{1^n : n \in \mathbb{N}\}$ 的特殊情况。

非均匀版本

计算不可区分的非均匀定义可通过人为扩充定义 C.5 中的分布索引得到。也就是说， $\{X_\alpha\}_{\alpha \in S}$ 和 $\{Y_\alpha\}_{\alpha \in S}$ 在非均匀意义上是计算不可区分的，如果对所有多项式 p ，概率空间 $\{X'_{\alpha'}\}_{\alpha' \in S'}$ 和 $\{Y'_{\alpha'}\}_{\alpha' \in S'}$ 是计算不可区分的（如定义 C.5 给出），其中对所有 $\beta \in \{0,1\}^{p(|\alpha|)}$ ， $S' = \{\alpha\beta : \alpha \in S \wedge \beta \in \{0,1\}^{p(|\alpha|)}\}$ ，并且 $X'_{\alpha\beta} = X_\alpha$ （ $Y'_{\alpha\beta} = Y_\alpha$ ）。由定义 8.12 中的公式可以得到一个等价的定义。

C.3.2 伪随机数发生器

简单地说，伪随机数发生器是一个高效的（确定性）算法，输入短的随机种子，输出一个与均匀选择的序列在计算上不可区分的较长（长得多）序列。

定义 C.6 (伪随机发生器，重申定义 8.1) 令 $l: \mathbb{N} \rightarrow \mathbb{N}$ 满足 $l(n) > n$ ，对所有 $n \in \mathbb{N}$ ，带有扩张函数 l 的伪随机发生器是一个满足如下条件的（确定性）多项式时间算法 G 。

(1) 对所有 $s \in \{0,1\}^*$ ，有 $|G(s)| = l(|s|)$ 。

^① 为使定义 C.5 和定义 C.4 自然过渡（不像定义 8.4），这里的区分器可以得到分布的指数（还是索） n （在典型的应用中，定义 8.4 和定义 C.4 的不同在于从相应分布的任意样本易于确定指数 n ）。

(2) $\{G(U_n)\}_{n \in \mathbb{N}}$ 和 $\{U_{l(n)}\}_{n \in \mathbb{N}}$ 是计算不可区分的, 其中 U_m 指 $\{0,1\}^m$ 上的均匀分布。

事实上, 概率空间 $\{G(U_n)\}_{n \in \mathbb{N}}$ 称为伪随机的。

需要强调的是, 不仅在“标准”的算法应用中可用伪随机序列代替真随机序列, 在密码学算法中也可以。换言之, 任何合法用户使用真随机序列时安全的密码学应用, 在使用伪随机序列时仍为安全的。这种代替(伪随机序列代替随机序列)的好处是, 后一种序列能够通过很少的真随机数高效生成。在交互式场合中, 通过使用离线(或在初始化阶段)选择的固定随机种子生成的伪随机比特, 可以消除在线执行中所有的随机步骤。这就允许交互双方基于一个预先共享的随机数种子生成可以共用的较长秘密比特串。

伪随机数发生器的各种密码学应用系统将在后面给出, 但是我们首先需回顾“伪随机数发生器存在当且仅当单向函数存在”(见定理 8.11)。对伪随机数发生器的进一步讨论可见 8.2 节。

C.3.3 伪随机函数

伪随机函数提供了一种由短的随机种子产生长的伪随机序列的方法。Goldreich、Goldwasser 和 Micali^[95]引入并构造的伪随机函数拥有更强的能力: 能够对大的(逐比特扫描不可行的)伪随机序列提供对比特位的高效直接访问。准确地说, 伪随机函数是高效的(确定性)算法, 给定一个 n 比特的种子 s 和 n 比特的参数 x , 能返回一个 n 比特字符串, 记作 $f_s(x)$, 使得对均匀选择的 $s \in \{0,1\}^n$, 要区分真随机函数 $F: \{0,1\}^n \rightarrow \{0,1\}^n$ 的值与 f_s 的值是不可行的。即对(可行的)测试程序给定对函数的预言访问(但不是函数的精确描述), 它不能区分对伪随机函数的预言访问和对真随机函数的预言访问。

定义 C.7 (伪随机函数) 伪随机函数(空间)是满足下列条件的一组函数集合 $\{f_s: \{0,1\}^{|s|} \rightarrow \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$ 。

(1) (高效计算) 存在高效的(确定性的)算法, 对给定的种子 s 和参数 $x \in \{0,1\}^{|s|}$, 返回 $f_s(x)$ 。

(2) (伪随机性) 对所有概率多项式时间预言机 M 、所有正多项式 p 和所有足够大的 n , 有

$$|\Pr[M^{f_{U_n}}(1^n) = 1] - \Pr[M^{F_n}(1^n) = 1]| < \frac{1}{p(n)}$$

式中: F_n 表示均匀选择的从 $\{0,1\}^n$ 到 $\{0,1\}^n$ 的函数映射。

上述定义的主要特征是, 可以仅仅通过生成并共享种子的方法来生成并共享伪随机函数, 即一个“看上去随机”的函数 $f_s: \{0,1\}^n \rightarrow \{0,1\}^n$, 由它的 n 比特种子 s 确定。各参与方为了共享一个“看上去随机”的函数 f_s (判定 2^n 更多的值), 仅需要产生和共享 n 比特的种子 s (例如, 一方可以随机选择种子 s , 并通过安全信道向所有其他方发送)。共享伪随机函数使得参与方可以依据当前的具体环境(不需要事先了解)来确定(自行产生且不需要任何进一步的通信)看上去随机的值。为充分发挥这个工具的潜力, 要认识到共享随机函数本质上等同于联机协商与(在线)给定值相关的随机值; 其中给定值从较大的可能的值集中取值。要强调, 这种协商不需要通信和同步: 当参与方要将随机值与一个给定值 $v \in \{0,1\}^n$ 相关联时, 它将要把与 v 相同的随机值 $r_v \in \{0,1\}^n$ 关联起来(通过设 $r_v = f_s(v)$, 其中 f_s 是事先约定的伪随机函数)。伪随机函数的具体应用出现在 C.5.2 节和 C.6.2 节中。

定理 C.8 (如何构造伪随机函数) 可以用任意随机数发生器构造伪随机函数。

证明思路:^①令 G 为一个种子由 2 的倍数 (即 $l(n)=2n$) 扩充生成的伪随机发生器, 令 $G_0(s)$ ($G_1(s)$) 表示 $G(s)$ 的前 (后) $|s|$ 个比特位。定义

$$G_{\sigma|s|} \cdots \sigma_2 \sigma_1(s) \stackrel{\text{def}}{=} G_{\sigma|s|}(\cdots G_{\sigma_2}(G_{\sigma_1}(s)) \cdots)$$

考虑函数空间 $\{f_s : \{0,1\}^{|s|} \rightarrow \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$, 其中 $f_s(x) \stackrel{\text{def}}{=} G_x(s)$ 。形象地说, 函数 f_s 定义为 n 在步内走过一棵深度为 n 、顶点带有标签的完全二叉树。树的根, 后面表示为树在第 0 层的顶点, 用字符串 s 作标记。如果一个中间节点被标记为 r , 则其左孩子标记为 $G_0(r)$, 而其右孩子标记为 $G_1(r)$ 。 $f_s(x)$ 的值是, 由根出发通过与字符串 s 相关的路径可达叶子节点上相应的字符串。

可认为函数空间 $\{f_s\}_{s \in \{0,1\}^*}$ 是伪随机的。利用混合技术可以证明 (见 8.2.3 节): 第 i 个混合 H_n^i , 是由定义在 $\{0,1\}^n \rightarrow \{0,1\}^n$ 上的函数 $2^{2^i n}$ 组成的函数空间, 每个函数由 2^i 个随机的 n 比特字符串 $\bar{s} = \langle s_\beta \rangle_{s \in \{0,1\}^i}$ 确定。函数 $h_{\bar{s}}$ 在 $x = \alpha\beta$ 的取值定义为 $G_\alpha(s_\beta)$, 其中 $|\beta|=i$ 。形象地说, 函数 $h_{\bar{s}}$ 由与第 i 层顶点对应的字符串, 以及用与 f_s 定义的规则来标记的较低层顶点来定义。两端的混合对应于不可区分性断言 ($H_n^0 = f_{U_n}$ 和 H_n^n 是真随机函数), 而相邻混合的不可区分性可由不可区分性假设 (利用可约性方法) 得到。特别地, 我们证明如果能区分 H_n^i 与 H_n^{i+1} , 则也可以区分 $G(U_n)$ 与 U_{2n} 的多个样本 (通过在第 $i+1$ 层的足够多结点上即时地替换一半给定样本)。□

变形

伪随机函数概念的有用变形 (和推广), 包括定义在所有字符串上 (即 $f_s : \{0,1\}^* \rightarrow \{0,1\}^*$) 的伪随机布尔函数, 以及定义在其他定义域和值域上 (对任意多项式界限函数 $d, r : \mathbf{N} \rightarrow \mathbf{N}$, $f_s : \{0,1\}^{d|s|} \rightarrow \{0,1\}^{r|s|}$) 的伪随机函数。这些变形之间的各种变换在 (见参考文献[91]中的 3.6.4 节) 和 (参考文献[92]中的附录 C.2) 中描述。

C.4 零 知 识

零知识证明为密码协议的设计提供了强大的工具, 也为研究关于这些协议的不同问题提供了一个良好的基准。大体上讲, 零知识证明不会泄露超出断言有效性的任何信息的证明。即获得证明的验证者仅能确信断言的有效性 (如同由第三方告知该断言是成立的)。可以认为, 任何由零证明计算得到的信息由 (有效的) 断言本身也能计算得到。后一种说法符合后面要讨论的仿真范式, 9.2.1.1 节中讨论了这种范式在密码学中的应用。

C.4.1 仿真范式

对安全进行建模的关键问题是如何表达这一直观需求, 即敌手通过背离诚实用户的预设行为而在 “本质上没有任何收获”。对这一问题, 仿真范式回答是不能让敌手得到任何信息, 无论敌手从不受限制的攻击行为能得到什么, 仅通过良性行为也能有本质上相同的收获。“良

① 细节见参考文献[91]中的 3.6.2 节。

性行为”的定义解释了安全性需求，对于想要解决的安全问题也比较具体。如在零知识证明中，不受限制的攻击行为可由一个任意概率多项式时间的验证策略描述，无论良性行为是基于（仅）断言本身的何种计算（假定断言是有效的）。在 C.5.1 节和 C.7.1 节中讨论了其他的例子。

可以认为，仿真范式中定义安全的方法（以及本附录中整体上采用的定义方法）有点过于谨慎，因为其中也禁止某种“莫须有”的敌手的“无害”收获。^①对这种印象我们要提出警告。首先，在密码学没有什么比考虑“合理的”敌手（一个几乎对立的观念）更加危险的：敌手会尽力去尝试那些系统设计者认为是“莫须有”的恶意行为。其次，似乎无法给出这样的安全性定义，能够区分“以有害的方式破坏一个系统”和“以无害的方式破坏一个系统”：是否有害取决于具体应用，而一个良好的安全性定义应该与应用无关（否则，在任何新的应用系统中使用该密码方案时都需要重新评估其安全性）。进一步地，即使是在特定的场合，也非常难以界定“有害的攻击”。

C.4.2 确切定义

9.2.1.2 节中将零知识定义为证明者策略的某种属性（在交互式证明系统的上下文中，如在 9.1.1 节中所定义）。更一般地，该术语可以应用于任何交互式机器，而无需考虑其目标是什么。如果对所有可行的策略 B^* ，存在可行的计算 C^* 使得如下两个概率空间在计算上不可区分（根据定义 C.5），则称策略 A 对集合 S 是零知识的。

(1) $\{(A, B^*)(x)_{x \in S}\}$ 在与 A 对相同输入 $x \in S$ 进行交互后 B^* 的输出。

(2) $\{C^*(x)_{x \in S}\}$ C^* 对于输入 $x \in S$ 得到的输出。

第一个空间表示一个交互式协议的真正执行过程，而第二个空间表示不与任何人交互的独立进程（称为“仿真器”）的计算。

上述定义没有考虑敌手 B^* 在进行交互前拥有的附加信息。对于将零知识证明用作子协议的更大协议来说，这些附加信息很重要。为此，提出一个更严格的称为附加输入零知识（Auxiliary-input Zero-knowledge）的定义，9.2 节中未包含这个概念。

定义 C.9（零知识，回顾） 如果对任意概率多项式时间策略 B^* 和任意多项式 p ，存在一个概率多项式时间算法 C^* ，使得如下两个概率空间在计算上不可区分，则称策略 A 关于 S 中输入是附加输入零知识的。

(1) $\{(A, B^*(z))(x)_{x \in S, z \in \{0,1\}^{p(|x|)}}\}$ 在拥有附加输入 z 时与 A 对相同的输入进行交互后 B^* 的输出。

(2) $\{C^*(x, z)_{x \in S, z \in \{0,1\}^{p(|x|)}}\}$ 为 C^* 对于输入 $x \in S$ 和 $z \in \{0,1\}^{p(|x|)}$ 得到的输出。

几乎所有已知的零知识证明都是附加输入零知识的。前面有所暗示，附加输入零知识性在顺序组合下可以保持。通过重复调用给定会话中关于敌对验证者策略的单会话仿真器，可以构造多会话协议仿真器（将前一个会话的描述作为仿真器的输入）。事实上，剩余单会话验证者得到的描述就是附加输入的一部分（即定义 C.9 中的 z ）。细节见参考文献[91]中 4.3.4 节。

^① 事实上，关于仿真范式，只有当所有可能的敌手都能被充分的良好行为模仿时，才称一个系统是安全的。因此，该方法还考虑“莫须有”的敌手，且并不回避无法模仿的“无害”收获。

C.4.3 一般性结论及应用

目前为止,没有涉及的问题是零知识证明的存在性。显然, $\mathcal{P}$ (或者 BPP) 中的每个集合都有一个“简单的”零知识证明(其中由验证者自己判定成员关系);然而,我们所寻求的是对验证者自身无法确定的断言的零知识证明。

假定“承诺方案”(见 4.3.1 节)存在,如果单向函数存在,便可推出承诺方案存在^[118, 169],则对任意 NP-集存在成员关系的(附加输入)零知识证明。构造 9.10 中抽象地描述了这类零知识证明,它们有如下重要属性:倘若以(已被证明的)断言的 NP-证据作为附加输入,则既定证明者策略是高效的。^①事实上,通过标准的到 3-着色问题的 Karp 归约,利用构造 9.10 中的协议可得到对任意 NP-集的零知识证明。(构造 9.10 中的)抽象盒子可利用承诺方案实现,从而得到:

定理 C.10 (零知识证明的适用性 (定理 9.11, 重述)) 如果(非均匀困难的)单向函数存在,则每个集合 $S \in NP$ 都有一个附加输入零知识交互证明。而且,倘若在 S 的公共输入中,将成员关系的 NP-证据作为给定附加输入,则既定证明者策略就能在概率多项式时间内实现。

定义 C.10 使零知识证明成为密码方案和协议设计中的有力工具(见 C.4.3.3 节)。在定理 C.10 中困难假设非常关键。

C.4.3.1 承诺方案

简单地说,承诺方案是一个两阶段(两方)协议,其中允许一个参与方对某个值做出承诺(在第一阶段),并将该值保密。在第二阶段,将承诺“公开”,并且要保证这种“公开”仅产生一个在承诺阶段产生的值。于是,承诺方案(的第一阶段)既有绑定(Binding)也有隐藏(Hiding)。

基于任意单向 1-1 函数 f (其困难核心为 b)可以构造简单的(单向通信的)承诺方案。为承诺一个比特 σ ,发送方均匀选择 $s \in \{0,1\}^n$,发送一对 $(f(s), b(s) \oplus \sigma)$ 。注意:这是既绑定且隐藏的。另一个构造基于任意单向函数,使用伪随机数发生器 G ,其种子具有因数 3(见定理 8.11)。通过双向通信可按如下方法实现承诺(见参考文献[169]):接收方均匀选择 $r \in \{0,1\}^{3n}$ 且将其传给发送方,发送方均匀选择 $s \in \{0,1\}^n$,并且要承诺 1 时发送 $r \oplus G(s)$,要承诺 0 时发送 $G(s)$ 。关于绑定性,可以看到对于一些 (s_0, s_1) 至多有 2^{2n} 个“坏”的 r 满足 $G(s_0) = r \oplus G(s_1)$,并且接收者将以压倒性的高概率选不到坏值。隐藏属性可由 G 的伪随机性得到。

C.4.3.2 效率方面的考虑

协议的轮数通常是最重要的效率标准(或复杂性尺度),而且典型地希望它为常量。然而,为获得可忽略的合理错误,构造 9.10 中的协议必须是非常数时间的(对所得到的协议的分析依赖在顺序组合下零知识性的保持性)。看上去,通过并行(而不是串行)地完成这些调用,似乎可以衍生出一个常数轮的零知识证明系统(具有可忽略的合理错误)。不幸的是,不清楚所得到的交互式证明是不是零知识的。而且,在标准的困难假设下(如因子分解的困难性),常数轮零知识证明(具有可忽略的合理错误)在 NP 的每个集中都存在。

^① 对既定证明者的附加输入(为允许)不能与对给恶意验证者(在附加输入零知识定义中)的附加输入相混淆。前者是对公共输入的 NP-证据,由证明者策略(参见 C.4.3.3 节讨论的普遍应用)中涉及的用户可得到。相比之下,恶意验证者的附加输入模仿了敌手可得到的任意部分信息。

C.4.3.3 一般应用

如前所述, 定理 C.10 使零知识成为密码方案和协议设计中的一个强大工具。定理 C.10 的两个重要方面拓宽了零知识的应用: 首先, 定理 C.10 为每个 NP-集提供了一个零知识证明; 其次, 当给定充分的 NP-证据时, 既定的证明者可在概率多项式时间内实现。现在转向零知识证明的典型应用。

在典型的密码学场景中, 用户 U 拥有一个秘密并基于该秘密完成一些操作。问题在于其他用户如何验证 U 确实完成了正确的操作 (由 U 的秘密和公开的已知信息判定)。实际上, 如果 U 泄露了该秘密, 则任何人可以验证 U 完成了正确操作。然而, U 并不想泄露秘密。零知识证明可以满足这两个冲突的需求 (既让其他用户验证 U 完成了正确的操作, 也不会因泄露秘密损害 U 的利益)。也就是说, U 能够在零知识中证明确实完成了正确的操作。注意: 称 U 完成了正确操作是一个 NP-断言 (因为 U 的合法操作是由关于秘密的多项式时间函数和公开信息确定的), 且 U 拥有其有效性的 NP-证据 (即秘密是其行为符合公开信息的 NP-证据)。于是, 通过定理 C.10, U 可以有效地验证其操作的正确性而不会泄露关于其秘密的任何信息。相应地, 要求 U 证明 (在零知识下) 其行为的合法性是公平的。“强制性合法行为” 是零知识证明的经典应用 (见 C.7.3.2 节)。

这种模式 (即通过零知识证明的“强制性合法行为”) 可由定理 C.10 得到, 且已应用于大量不同场景中。事实上, 这种模式是零知识协议在密码学中广泛应用的基础。

C.4.4 其他定义及相关概念

本节要考虑零知识证明概念的许多变形和其所依赖的交互式证明模型, 包括黑盒仿真和零知识证明的其他变形 (参考 C.4.4.1 节), 如知识证明、非交互式零知识、证据不可区分证明的概念等 (参考 C.4.4.2 节)。

首先请注意 C.9.1.4.2 节讨论的计算合理的概念和论据系统的相关概念。论据系统可能比交互式证明系统和提供较强的零知识保证更高效。特别是, 能够基于任何单向函数来构造对 $\mathcal{NP}$ 类的几乎完美的零知识论据^[172], 其中几乎完美的零知识意味着仿真器的输出在统计上接近于在真实交互中验证者的观察 (见 C.4.4.1 节中的讨论)。注意: 对证明者更强的安全性保证 (如几乎完美零知识所给出的) 是以对验证者较弱的安全性保证 (如计算合理性所给出的) 为代价的。对于这种折中是否值得的问题似乎与应用无关, 应该考虑相应协议的可用性和复杂性。

C.4.4.1 其他定义

考虑几个关于零知识概念 (如定义 C.9 所定义的) 的其他定义。

全局性与黑盒仿真

通过要求存在一个全局仿真器 C , 可以得到定义 C.9 的加强版本, 当给验证者程序一个附加输入时, 该仿真器能够仿真 (交互式地获得) 任意验证者策略 B^* 。也就是说, 依据定义 C.9, 应该用 $C(x, z, \langle B^* \rangle)$ 替代 $C^*(x, z)$, 其中 $\langle B^* \rangle$ 指 B^* 的程序的描述 (可能依赖于 x 和 z)。即通过要求它是关于验证者程序的 (而不是依赖于它的任意的) 均匀 (可行) 函数, 可以有效地约束仿真。这种约束很正常, 因为很难想象通过其他方法建立给定协议的零知识属性。进一步地, 可以认为由于对程序采取逆向分析是不可行的, 仿真器在仅使用验证者策略时将与预言机一样 (或如同一个“黑盒”)。由此引出黑盒仿真的概念, 参考文献[98]引入并讨论

了这个概念，还有大量的工作也对其进行更深入的研究。一种观点相信黑盒仿真先天的局限性也带来了零知识本身的先天局限性。例如，认为不能用黑盒模式来仿真构造 9.10 中交互式证明的并行版本（除非 NP 包含于 BPP），这意味着该版本不是零知识的（如定义 C.9 本身）。然而，Barak^[25]最近反驳了任意零知识协议能在黑盒模式下仿真的观点。

诚实验证者与一般的欺骗性验证者

定义 C.9 涉及所有可行的验证者策略，在密码学中这很自然，因为零知识性刻画了证明者在面对任何想通过与其交互而有所收获的可行（即敌对的）努力情况下的健壮性。一个较弱但仍然有意义的零知识概念涉及到与证明者直接交互的“诚实验证者”（甚至一个半诚实验证者）^①能获得什么，除了它保留（并输出）整个交互记录（即使要其销毁所有交互记录）之外。尽管这种较弱概念不满足标准的密码应用系统，但从概念和复杂性理论角度看它也是很吸引人的。而且，每个有诚实验证者的零知识证明系统都能转换为标准零知识证明（不需要困难假设，且在“公开掷币”情况下证明这样做不会明显增加证明者的计算代价，见参考文献[228]）。

统计零知识与计算零知识

回顾定义 C.9 假设对每种类型的概率空间（即表示在与证明者交互后验证者的输出），存在另一种类型（即表示仿真器的输出）的“类似”空间。其中一个关键参数是“相似性”，文献中对其有 3 种解释，从而得到零知识的 3 种概念。

(1) 完美零知识，要求两种概率空间是同分布的。^②

(2) 统计（或几乎完美）零知识，要求两个概率空间在统计上接近（即它们的距离可忽略）。

(3) 计算（或普通）零知识，要求概率空间是计算不可区分的。

事实上，计算零知识是最自由的概念，也是定义 C.9 中的概念。注意到，具有统计零知识证明的问题类中包含几个困难问题。感兴趣的读者可见参考文献[227]。

C.4.4.2 相关概念：POK、NIZK 和 WI

简要讨论知识的证明(POK)、非交互零知识(NIZK)以及证据不可区分证明(WI)的概念。

知识证明

粗略地讲，知识证明是交互式的证明，其中证明者宣称其拥有某个对象的“知识”（如图的 3-着色方案），而不仅仅是其存在性（如存在图的 3-着色方案，可直接得到图的 3-可着色性这一断言）。进一步的讨论见 9.2.3 节。我们指出，“知识证明”，特别是零知识的“知识证明”在密码方案和协议的设计中有很多应用。关于知识的零知识证明的一个著名应用是构造认证方案（如 Fiat-Shamir 方案）。

非交互零知识

非交互零知识证明系统模型由 3 个实体组成：证明者、验证者以及均匀选择的参考字符串（视为由可信第三方选择）。验证者和证明者都要读取参考字符串（也可以读共同输入），每一方都可以运行附加的掷币算法。交互中包含证明者发给验证者的消息，验证者最后得出

① 当考虑定义 C.9 的另一种形式时，“半诚实验证者”的术语十分吸引人，在另一种定义（见参考文献[91]中 4.3.1.3 节）中，只要求仿真器生成验证者在真实交互中的观察，而验证者的观察中包含了其（公共及辅助的）输入、随机数以及收到的所有信息。

② 完美零知识证明的真正定义允许仿真者以可忽略的概率失败（在输出一个特定符号时），并且仿真的输出分布由它未失败的情况决定。

结论（是否接受共同输入）。（基本的）零知识需要用到一个仿真器，输出一对在真实模型中的分布（这对值由均匀选择的参考字符串和随机植入的消息组成）在计算上不可区分的值。^①非交互式零知识证明系统有许多应用（如构造公钥加密和签名方案，其中参考字符串可能包含于公钥中）。文献中给出了关于非交互式零知识证明的几种不同定义（见参考文献[91]中4.10节和参考文献[92]5.4.4.4节）。可基于标准的困难性假设（如分解整数的困难性）来构造任意NP集的NIZK证明，但是即便是构造基本的NIZK证明系统也似乎比构造交互式零知识证明难得多。

证据不可区分性

作为对零知识概念有意义的弱化，参考文献[76]中提出了证据不可区分性的概念。粗略地讲，对任意NP-关系 R ，如果不存在可行的验证者能够区分证明者使用一个对 x 的NP-证据（即满足 $(x, w_1) \in R$ 的 w_1 ）的情形，和证明者使用另一个对 x 的不同的NP-证据（即满足 $(x, w_2) \in R$ 的 w_2 ）的情形，则对相应NP-集的证明（或论证）系统是证据不可区分的。显然，任何零知识协议都是证据不可区分的，但反之不一定成立。而且，证据不可区分的协议似乎比零知识协议更易于构造。证据不可区分协议的另一个优势是，在任意并发组合下是封闭的，而即使在并行组合时（一般）零知识协议都不是封闭的。证据不可区分协议是构造更复杂协议的一个重要工具。特别值得一提的是，参考文献[75]中基于证据不区分证明（论说不上）构造零知识证明（和论证）的技术。

C.5 加密方案

在不安全的信道上实现保密通信是密码学中传统且最根本的问题。这个问题的环境中，通信双方通过一个可能被敌手窃听的信道进行通信。双方希望交换信息，但是又想让“搭线窃听者”尽可能看不懂得到的信息。这个问题的经典解决方法是使用加密。简单地说，加密方案是使得各方可以进行秘密通信的协议。加密方案一般包括一对算法：一个称为加密，由发送方（即发送消息的一方）执行；另一个称为解密，由接收方执行。于是，在发送消息之前，发送方首先对消息应用加密算法，将计算结果（称为密文）通过信道发送。收到密文后，另一方（即接收方）对其应用解密算法，恢复出原始的消息（称为明文）。

为了能提供安全的通信，接收方必须知道一些窃听者不知道的信息（否则，窃听者就能和接收者一样解密消息）。这些额外的知识可以是解密算法本身，或者是解密算法使用的一组参数和（或）附加的输入。这些额外的知识被称为解密密钥。注意：不失一般性，可以假定窃听者也知道解密算法，并且解密算法有两个输入：密文和解密密钥（这样描述的预设条件是存在有效的算法以生成（随机的）密钥）。我们强调解密密钥（为窃听者所不知）的存在仅仅是保密通信的一个必要条件。

评估加密算法的“安全性”是一项很复杂的任务。首先要理解什么是“安全性”（即恰当地定义这个直觉术语的含义）。已知有两种定义方法。第一种（“古典的”）是由Shannon引入^[205]的信息论方法。它关注的是密文中“提供”的关于明文的“信息”。简单地说，如果密文中包含关于明文的信息，则认为加密方案不安全。已经证明，为了达到如此高级别（即“完

^① 注意：在真实模型中验证者没有影响分布，所以零知识的基本定义不涉及它。在定义的适应性变形中涉及到了验证者（或者说是一个适应性地选择待证明断言的进程）。

善的”)的安全性,所使用的密钥长度至少要 and 加密的总消息长度一样长^[205]。这个事实(即密钥比要交换的信息还要长),严重限制了这种(完善安全的)加密方案的适用性。

第二种(“现代的”)方法基于计算复杂性。这种方法的主要观点是,密文是否包含关于明文的信息并不重要,重要的是,这些信息能否被有效提取。换句话说,不是要问窃听者提取特定的信息是否可能,而是问窃听者提取这个信息是否可行。利用这种新的方法(即“计算复杂性”方法),即使加密方案的密钥比处理的总消息的长度要短,也能提供安全性。

计算复杂性方法可以引入信息论背景下不存在的概念和原语。一个典型的例子是由 Diffie 和 Hellman^[66]提出的公钥加密方案(最流行的方案是由 Rivest、Shamir 和 Adleman^[193]提出的)。回顾一下前面讨论的重点是解密算法及其密钥。加密算法当然也必须要有除消息之外的、依赖于解密密钥的附加输入。这个附加输入称为加密密钥。传统的加密密钥,特别是 20 世纪 80 年代之前,长达 1000 多年中所使用的所有加密方案,都使用了与解密密钥相同的加密密钥。于是,这些方案中的窃听者绝不能知道加密密钥,由此产生了密钥分发问题,即通过不安全信道进行通信的双方如何协商一个秘密的加密/解密密钥(传统的解决方法是用另一个安全信道来交换密钥,尽管这样做代价很大)。计算复杂性方法使得有可能引入这样的加密方案,其中窃听者可以知道加密密钥,然而不会影响安全性。显然,这种方案中的解密密钥和加密密钥不同,而且从加密密钥中获得解密密钥是不可行的。这种加密方案被称为公钥加密方案,其优势在于解决了密钥分发问题(因为加密密钥可以公开)。换言之,一旦 X 产生一对密钥并公开加密密钥,则任何一方都可以给 X 方发送加密的消息, X 方能够获得真正的信息(即明文),而任何其他人都不能得到关于明文的任何信息。

和公钥方案相比,传统的使用相同加密密钥和解密密钥的加密方案称为私钥加密方案,这些方案中的加密密钥必须保密(而不像公钥方案中一样可以公开)。需要指出,任何方案的完整描述中都需要定义密钥生成算法,即对于给定的安全参数,能产生一对(随机的)加密、解密密钥(在私钥加密方案中两者相同)。

于是,私钥和公钥加密方案都包含 3 个高效算法:密钥生成算法,记作 G ; 加密算法,记作为 E ; 解密算法,记作 D 。对每一对由 G 生成的加密、解密密钥 (e, d) 和每一个明文 x , 有 $D_d(E_e(x)) = x$, 其中 $E_e(x) \stackrel{\text{def}}{=} E(e, x)$ 且 $D_d(y) \stackrel{\text{def}}{=} D(d, y)$ 。两种加密方案的不同也体现在安全性的定义中:公钥加密方案中即使敌手知道加密密钥时也能保证安全性,而私钥加密方案不要求这一点。以下的定义中主要讨论公钥加密情形(忽略敌手输入中的加密密钥就可以得到私钥情形)。

C.5.1 定义

高明的化妆不应泄露人的身高。

Shafi Goldwasser, Silvio Micali, 1982

为简单起见,首先考虑加密单个消息(还可以进一步简化,假定消息长度为安全参数 n)^①。由前述讨论得知,如果从密文(以及加密密钥)中获得关于明文的任何信息都是不可行的,则公钥加密方案是安全的。即由密文及某种先验信息中计算出的关于明文的任何信息,都可单独由这种先验信息高效求得。这个关于安全性的基本定义称为语义安全性,由

① 这个简化的假设并没有使公钥方案失去一般性,但私钥方案情形要考虑对多项式数量消息的加密(如同本节末的讨论)。

Goldwasser 和 Micali^[108]提出。

定义 C.11 (语义安全) 公钥加密方案 (G, E, D) 为语义安全的条件是: 如果对所有的概率多项式时间算法 A , 存在一个概率多项式时间算法 B , 对任意两个函数 $f, h: \{0, 1\}^* \rightarrow \{0, 1\}^*$ 和所有的概率空间 $\{X_n\}_{n \in \mathbb{N}}$, 满足 $|h(x)| = \text{poly}(|x|)$ 和 $X_n \in \{0, 1\}^n$, 并有

$$\Pr[A(e, E_e(x), h(x)) = f(x)] < \Pr[B(1^n, h(x)) = f(x)] + \mu(n)$$

式中: 明文 x 的分布服从 X_n ; 加密密钥 e 的分布服从 $G(1^n)$; $\mu(n)$ 是一个可忽略的函数。

也就是说, 要成功地从 $h(x)$ 推断出 $f(x)$, 与从 $h(x)$ 和 $(e, E_e(x))$ 推断出 $f(x)$ 的难度是一样的, 这也意味着从 $(e, E_e(x))$ 中不能获得任何信息。注意, 对函数 h 和 f 没有计算上的约束。我们强调, 前述定义 (后一个也是) 主要指公钥加密方案, 对算法 A 不提供公钥 e 时就是私钥方案。

以下对安全性的技术解释是: 区分任意两个明文 (长度相同) 加密后的密文是不可行的。^① 这个定义 (原出自参考文献[108]) 与定义 C.11 等价 (要实现它需要一个概率加密算法)。

定义 C.12 (加密的不可区分性) 公钥加密方案 (G, E, D) 具有加密不可区分性的条件是: 对任意概率多项式时间算法 A , 任意三元组序列 $\{x_n, y_n, z_n\}_{n \in \mathbb{N}}$, 其中 $|x_n| = |y_n| = n$, $|z_n| = \text{poly}(n)$, 有

$$|\Pr[A(e, E_e(x_n), z_n) = 1] - \Pr[A(e, E_e(y_n), z_n) = 1]| = \mu(n)$$

同样, 加密密钥 e 的分布服从 $G(1^n)$, $\mu(n)$ 是一个可忽略的函数。

特别地, $|z_n|$ 可能等于 (x_n, y_n) 。因而, 要区分任意两个固定消息 (如全 0 消息和全 1 消息) 的密文是不可行的。于是, 如下的格言仍有效:

高明的化妆不应使母亲认出自己的孩子。

Shafi Goldwasser, Silvio Micali, 1982

在许多以加密为终极目标的应用场合中, 定义 C.11 极具吸引力。定义 C.12 一方面用于证明候选加密方案的安全性, 另一方面也用于分析加密方案在密码协议中的应用。因而, 这两个定义的等价性是十分关键。

定义 C.11 和定义 C.12 的等价性——证明思路

直观上, 加密的不可区分性 (即对的加密) 是语义安全的特例。确切地说, 它所对应的情况是 X_n 在 $\{x_n, y_n\}$ 上是均匀的, 函数 f 指示一个明文, h 不能区分这些明文 (即如果 $w = x_n$ 且 $h(x_n) = h(y_n) = z_n$, 则 $f(w) = 1$, 其中 z_n 的定义见 C.12)。另一个方向的证明考虑算法 B , 输入为 $(1^n, v)$, 其中 $v = h(x)$, 算法 B 生成 $(e, d) \leftarrow G(1^n)$ 并输出 $A(e, E_e(1^n), v)$, 其中 A 如定义 C.11。用加密的不可区分性来证明 A 与 B 相同 (即对所有 h, f 和 $\{X_n\}_{n \in \mathbb{N}}$, 有

$$|\Pr[B(1^n, h(X_n)) = f(X_n)] - \Pr[A(e, E_e(1^n), h(X_n)) = f(X_n)]|$$

近似等于 $\Pr[A(e, E_e(X_n), h(X_n)) = f(X_n)]$)。

概率加密

安全的公钥加密方案必须使用概率的 (即随机的) 加密算法, 否则, 给定加密密钥作为 (附加的) 输入, 要区分对全 0 消息的加密和对全 1 消息的加密是很容易的。^② 这就解释了健壮的安全性定义和概率化的加密 (Probabilistic Encryption) 方法之间的联系, 这种联系可以追溯

① 事实上, 为满足这个条件必须使用概率加密算法。

② 在私钥加密方案中这也成立, 其中考虑加密几个消息 (而不是上文中的单个消息) 的安全性。例如, 如果使用确定性加密算法, 则敌手可以根据一对不同消息的密文区分同一消息的两个密文。

到 Goldwasser 和 Micali 的开拓性工作^[108]。

进一步的讨论

我们强调, (等价的) 定义 C.11 和定义 C.12 的发展已经远远超越了从密文恢复明文的不可能性。后者其实是安全加密方案的最小条件, 但远不是充分条件。典型地, 加密方案在应用中, 即使获得明文的部分信息也可能会影响的整个应用系统的安全性。当设计与应用系统无关的加密方案时, 设计者并不知道哪部分信息会影响系统安全性, 哪部分不会。而且, 即使要针对具体应用环境定制一个加密方案, 也几乎不可能了解所有可能影响系统安全性的部分信息的特征。于是, 只能要求从密文获取关于明文的任何信息都是不可行的。在大部分应用环境中明文并不是均匀分布的, 敌手可能会得到一些先验信息。我们要求即使在这种情况下也要保证所有部分信息的保密性。也就是说, 即使提供了一些关于明文的先验信息, 也不能从密文中获得关于明文的任何 (新的) 信息 (超出能从明文的先验信息中所获得到的)。所有这些情况在语义安全的定义中都是合理的。加密的不可区分性的等价定义可用于证明候选方案的安全性, 以及讨论密码方案作为应用协议的组成部分时所产生的影响。

多个消息的安全性

定义 C.11 和定义 C.12 指的是加密方案用来加密单个消息时的安全性 (每个密钥)。由于明文可能比密钥长,^① 这些定义已经是非平凡的, 满足这些定义的加密方案 (即使在私钥模型中) 蕴含了单向函数的存在性。在许多情况下, 要用相同的加密密钥来加密许多明文。在用相同的加密密钥来加密多项式数量的明文时, 如果定义 C.11 (定义 C.12) 的条件成立, 则加密方案在多消息场合下是安全的 (见参考文献[92]中的 5.2.4 节)。在公钥模型中, 单消息场合下的安全性隐含着多消息场合下的安全性。要强调的是, 这在私钥模型中未必正确。

C.5.2 构造方法

实际应用中, 习惯上使用“伪随机发生器”作为私钥加密方案的基础。要强调的是, 如果“伪随机发生器”很容易被预测 (如“线性同余发生器”), 则这种方法非常危险。然而, 使用伪随机发生器 (定义 C.3.2 所描述的) 时这种实践方法是很合理的。下面给出一种更灵活的构造方法。

基于伪随机函数的私钥加密方案

我们提出一种简单构造方法, 其中使用 C.3.3 节中定义的伪随机函数。密钥生成算法是 (伪随机) 函数 f_s , 均匀选择其输入 $s \in \{0,1\}^n$ 作为种子。为了加密消息 $x \in \{0,1\}^n$ (使用密钥 s), 加密算法均匀地选择一个字符串 $r \in \{0,1\}^n$, 并产生密文 $(r, x \oplus f_x(r))$, 其中 $\oplus$ 表示比特串的异或操作。为了解密消息 (r, y) (使用密钥 s), 解密算法只需计算 $u \oplus f_x(r)$ 。该加密方案的安全性证明包括两步 (依照附录 C.3.3 中给出的一般方法论)。

(1) 证明这个方案的一个理想版本是安全的, 其中使用均匀选择的函数 $F: \{0,1\}^n \rightarrow \{0,1\}^n$, 而不是伪随机函数 f_s 。

(2) 得到结论: 真实的方案是安全的 (否则, 就能区分伪随机函数与真随机函数)。

注意: 如果从历史的角度看加密算法, 可以不要求随机性 (在加密过程中)。让所有参与方都使用相同的密钥 (用于加密) 来完成加密操作 (通过维护一个联合状态 (如计数器))。

① 回想为了简单起见, 我们只考虑长度为 n 的消息, 但是一般定义针对着任意长度 (n 的多项式) 的消息。要指出的是, 在定义 C.11 的一般形式中, 应该提供消息长度作为两个算法 (A 和 B) 的辅助输入。

实际上,当使用私钥加密方法时,普遍的情况是仅在两个特定方的通信中使用相同的密钥,在通信过程中更新联合状态。如果加密方案用于两个用户间的 FIFO 通信,并且能可靠地维护计数器状态值,则没必要发送计数器值(这样的方案与实践普遍使用的“流密码”有关)。

C.5.3 超越窃听的安全性

我们的讨论只涉及到了“被动”攻击,其中敌手仅在密文传输线的路上进行窃听。很多应用场合中可能会出现更强的攻击(即“主动”攻击),最强的一种就是所谓的选择密文攻击。特别是在某些情况下,敌手可以让发送方加密由他选择的消息,有时候敌手甚至可以让接收方解密他所选择的密文。这就分别产生了选择明文攻击和选择密文攻击,附录 C.5.1 和附录 C.5.2 的安全性定义中未涉及到这样两种攻击。本小节主要讨论“主动”攻击,重点是选择密文攻击(最强的类型为“后验的”或“CCA2”)。

简单地说,在选择密文攻击中,敌手可以按自己的意愿得到对密文的解密结果,如果他能够获得与不同密文相对应的明文的相关信息,则认为攻击成功(见参考文献[92]中的 5.4.4 节)。换言之,敌手得到了与正使用的解密密钥相关的解密函数的预言机访问(在私钥方案情况下,也给出了相应加密函数的预言机访问)。敌手可以向预言机询问除“测试密文”(即)之外的任何密文。它还可以发起与合法密文无关的询问,也会得到相应的应答(即特殊的“失败”符号)。而且,敌手还可以影响测试密文的选择(指定一个分布,相应的明文要从中选择)。

能够抵抗选择密文攻击的私钥和公钥加密方案,能在构造相应被动方案使用的假设下构造出来。具体描述如下:

定理 C.13 假设单向函数存在,则存在选择密文攻击下安全的私钥加密方案。

定理 C.14 假设存在增强^①的陷门置换,则存在选择密文攻击下安全的公钥加密方案。

为证明这两个定理,可以构造相应的加密方案,使得敌手通过选择密文攻击而获得某种有用信息是不可行的。对于私钥加密方案(即定理 C.13),敌手产生合法密文是不可行的(除非作为对其选择的明文的加密预言应答而预先给定)。为实现这一点,可以在密文中加入一个“认证标签”,当解密密钥未知时,构造该标签是困难的。也就是说,使用消息认证方案(定义见 C.6 节)。对于公钥加密方案(即定理 C.14),敌手能自行生成密文,从而设计目标是使其在不“知道”相应明文时无法生成合法密文。为此,需要给明文加入关于相应明文的非交互的零知识“知识证明”。

选择密文攻击下的安全性与加密方案的不可延展性有关。简单地说,在不可延展的加密方案中,敌手生成与给定明文相关的另一个明文的密文是不可行的(如对已知的 x ,给定关于明文 $1x$ 的密文,要产生关于明文 $0x$ 的密文是不可行的)。进一步的讨论见参考文献[92]中的 5.4.5 节。

C.6 签名和消息认证

数字签名和消息鉴别都是保证数据有效性的方法,即验证数据是由特定的一方(或多个用户)提供的。签名方案和消息鉴别的区别在于,签名能够被“普遍验证”,而用户验证仅需

① 不严格地,增强中涉及到定义 C.2 中的困难性条件,并要求在给定用于采样 y 的随机数时,求 $f_i^{-1}(y)$ 也是困难的(见参考文献[92]附录 C.1)。

要的鉴别标签，也能由他们产生。

签名方案

数字签名（见参考文献[66, 182]）是在商业环境中计算机通信而产生的应用需求（其中，参与方需要做出承诺，并且（或者）申明自己所做的事）。对“不可伪造签名”的讨论甚至出现得早于计算时代，讨论的对象是手写签名（不是数字化的），所讨论的内容也和密码学没有关系。粗略地说，不可伪造签名方案应当满足如下条件：

- 每个用户都能对他所选择的消息高效地生成自己的签名。
- 每个用户都能高效地验证给定的字符串是否为另一用户对特定消息的签名。
- 想要对另一个用户没有签名的文件生成签名是不可行的。

注意到，不可伪造数字签名的描述清晰地体现了手写签名的基本要素。这些要素包括：每个人签署自己名字的能力，一个普遍认可的验证过程，对伪造签名不可行（或至少很困难）的确信（即对一份未签名文件生成其他人的签名，且该签名能被验证过程接受）。

消息认证方案

消息认证关注的是与加密方案相关的场景，即在不安全的信道上通信。此处主要考虑监听信道并可能篡改消息的主动攻击者。通过不安全信道进行通信的各方希望能够认证发出的消息，因而对方能辨别出原始的消息（发自发送者）和被篡改的消息（即由敌手篡改的）。简单地说，消息认证方案应满足如下条件：

- 每个通信方都能对他所选择的任何消息高效地产生认证标签。
- 每个通信方都高效地验证给定的字符串是否给定消息的认证标签。
- 外部的敌手（即除通信方之外的）想要为通信方没有发送的消息生成认证标签是不可行的。

注意：与数字签名方案的描述相比，此处不需要普遍验证：仅是指定的接收方需要验证认证标签。而且，不要求接收方无法生成认证标签（即仅要求局外人不能这么做）。因而，消息认证方案不能使第三方确信确实是发送方发出了信息（而不是接收方自己产生的）。相比之下，使用签名就能使第三方确信：将对文件的签名发送给另一方，以便他在以后可以使第三方相信（仅提供签名文件）文件的确是由签名者生成的（发出的或认可的）。

C.6.1 定义

签名方案和消息认证方案都由 3 个算法组成：密钥生成、签名和验证。与加密方案一样，密钥生成算法，记作 G ，产生一对相对应的密钥，一个用于签名（通过算法 S ），另一个用于验证（通过算法 V ）。也就是说，用 $S_s(\alpha)$ 表示由算法 S 对输入的签名密钥 s 和文档 α 生成的签名， $V_v(\alpha, \beta)$ 表示验证算法 V 对文件 α 和声称与验证密钥 v 相关的签名 β 得出的验证结论。毋庸置疑，对由 G 生成的任意密钥对 (s, v) 和所有的 α ，有 $V_v(\alpha, S_s(\alpha)) = 1$ 成立。

两种方案的不同体现在安全性定义中。在签名方案中，敌手可以得到验证密钥，而在消息认证方案中验证密钥（可能与签名密钥相同）不能交给敌手。因此，消息认证方案可看成是私钥版的签名方案。这个不同也导致功能上的不同（甚至不只是加密）：在签名方案的典型应用中，每个用户生成一对签名和验证密钥，将验证密钥公开，签名密钥保密。然后，每个用户用自己的签名密钥签署文件，这些签名能够被相应的公开验证密钥普遍地验证。相比之下，消息认证方案应用在一组相互信任，且协商过私钥的用户中，这个私钥用于生成和验证

认证标签。(实际上,在进行认证通信之前,已假定由相互信任的各方共同生成了私钥,或者通过安全的方式交换了私钥)。

我们主要讨论安全签名方案的定义,考虑能力很强的攻击者,并给予它很宽松的条件来攻击。特别是,攻击者可以获得对其任选的消息的签名。可能有人会认为,在许多环境中这种攻击是不可能的(原因是待签名的消息具有特定的格式)。然而,我们的观点是,不可能去定义通用的(即与应用环境无关)可接受消息的概念,因而似乎只能以这种方式来建立攻击的通用/健壮定义(注意到,即使在最坏的情况下,我们的方法也是极谨慎的)。类似地,如果能够对攻击过程中没有询问过的任何消息产生有效签名,则敌手获胜。同样,这意味着如果有能力对“无意义的”消息产生签名,也被视为方案被攻破。然而,没有办法建立通用的(即与应用环境无关)“有意义的”消息的概念(只有伪造对它们的签名才认为方案被攻破)。

定义 C.15 (安全的签名方案—概要) 选择消息攻击是这样一个过程,对输入的验证密钥,敌手可以得到所选择消息的签名(与签名密钥相关)。如果能够对攻击过程中没有询问过的消息输出有效的签名,则攻击成功(在存在性伪造意义上)。如果所有可行的选择消息攻击成功的概率都可忽略,则签名方案是安全的,其中概率出自初始选择的密钥对和敌手的攻击行为。

要强调的是,原始的 RSA (不同于 Rabin 方案和 DSS 的原始版本)根据以上定义是不安全的。然而,如果在使用 RSA (或其他方案)之前对其“随机化”,则有可能是安全的。

C.6.2 构造方法

使用伪随机函数(或附录 C.3.3 结尾讨论的推广形式的伪随机函数)可以构造安全的信息认证方案。具体地,密钥生成算法由为函数 $f_s: \{0,1\}^* \rightarrow \{0,1\}^*$ 选择种子 $x \in \{0,1\}^n$, (有效的)消息 x 关于密钥 s 的标签是 $f_s(x)$ 。与私钥加密方案一样,这种消息认证方案的安全性证明包括以下两步:

(1) 证明该方案的一个理想化版本是安全的(即不可伪造的),其中使用了均匀选择的函数 $F: \{0,1\}^* \rightarrow \{0,1\}^*$, 而不是伪随机函数 f_s 。

(2) 得到结论:真实的方案是安全的(否则,就能区分真随机函数和伪随机函数)。

注意到,这种消息认证方案“扩展了伪随机函数的应用”(即将伪随机函数直接应用于消息,其中需要伪随机函数的推广(见附录 C.3.3 结尾))。还可以更严格地使用伪随机函数,或者基于其他密码学原语来构造更高效的方案。

构造安全的数字签名方案似乎要比构造消息认证方案更困难。不过,也可以用同样的假设来构造安全数字签名方案。

定理 C.16 以下 3 个条件等价:

- (1) 存在单向函数;
- (2) 存在安全的数字签名方案;
- (3) 存在安全的消息认证方案。

要强调的是,与公钥加密方案不同,数字签名方案(可被视作消息认证的公钥密码模拟)的构造不要求陷门属性。

如何构造安全的签名方案

用于构造签名的 3 种主要范式是,“高效”签名密钥的“更新”、使用“认证树”以及“哈希范式”(后面要讨论)。除了在定理 C 证明的应用,这 3 个范式是相互独立的。

更新范式

参考文献[110]中介绍的更新范式，目的是降低选择密文攻击的潜在风险。其方法是这样的，用新生成（且随机）的签名方案实例来签署真正的消息，用固定的公钥来验证这个随机实例。^①在安全性方面直观来看，对固定的实例 (G, S, V) （与存放在公开文件中的固定验证密钥相关，并对攻击者是已知的）不能发起完整的选择消息攻击。攻击者可能获得的（通过对新方案的选择消息攻击）是签名，该签名与 (G, S, V) 的原始签名密钥 s 、随机字符串（遵循 $G(1^n)$ 的分布）及每个随机独立的签名密钥的附加签名相关。

认证树

与认证树相结合时，更新范式的安全性会提高。其主要思想是用公开验证密钥来认证几个（如两个）新的签名方案的实例，用每个实例来认证几个附加的新的实例，依次类推。于是，得到了树形的基本签名方案的实例，其中每个内部节点认证其子节点。用叶子节点表示对真正消息的签名，其中每个叶子至多签名一次。于是，每个真正消息的签名组成如下：

(1) 对该文件的签名，用与某个叶子节点相关的验证密钥来认证；

(2) 与从根到叶子的路径上节点相对应的一系列验证密钥，其中每个验证密钥分别由其父亲节点的验证密钥来认证。

要强调的是，与父亲节点密钥相关的同一签名可用于认证所有孩子节点的验证密钥。于是^②，签名方案的每个实例至多用于签署一个字符串（即实体居于中间节点时为一个验证密钥串，实体居于叶子时为真正的文件）。这种签名方案称为一次性签名方案，它比构造标准签名方案更容易，特别是在要签署的字符串比签名密钥（验证密钥）短很多时。例如，使用单向函数 f ，令签名密钥由 n 对字符串组成，相应的验证密钥为 f 的一系列映像，通过泄露适当的原像来签署 n 比特长的消息（即签名密钥为一系列 $((s_1^0, s_1^1), \dots, (s_n^0, s_n^1)) \in \{0, 1\}^{2n^2}$ ，相应的验证密钥为 $(f(s_1^0), f(s_1^1)), \dots, (f(s_n^0), f(s_n^1))$ ，对消息 $\sigma_1 \sigma_2 \dots \sigma_n$ 的签名是 $(s_1^{\sigma_1}, s_2^{\sigma_2}, \dots, s_n^{\sigma_n})$ 。）

哈希范式

注意：在前述的认证树中，签名方案的实例（与内部节点相关）用于签署一对验证密钥。于是，就需要一个一次性签名方案来签署比验证密钥更长的消息。为了桥接适用于短消息的（一次性）签名方案与可应用于较长消息的签名方案之间的缝隙，我们使用哈希范式方法。这种模式通过两阶段过程来签署文件：首先，将真正的消息散列成相对较短的字符串，然后使用基本签名方案签署这个字符串。如果哈希函数属于无碰撞（也叫抗碰撞）型，则这种方法是合理的。所谓无碰撞性，简单地说，就是给定一个从这类哈希函数族中随机选定的哈希函数，使用该哈希函数对两个不同字符串求得的散列值相同是不可行的。感兴趣的读者还可以参考用到了全局单向哈希函数这一看起来较弱概论的哈希范式变形（见参考文献[171]或参考文献[92]中的6.4.3节）。

① 也就是说，考虑一个基本签名方案 (G, S, V) ，用法如下。假定用户 U 生成了一对密钥 $(s, v) \leftarrow G(1^n)$ ，并将验证密钥 v 存放在一个公开文件中。当有人要求 U 来签署文件 α 时，用户 U 生成一个新（“新鲜的”）密钥对 $(s', v') \leftarrow G(1^n)$ ，用原来的签名密钥 s 签署 v' ，用新的签名密钥 v' 来签署 α ，将 $(S_s(v'), v', S_{v'}(\alpha))$ 作为对 α 的签名。通过检验 $V_v(v', \beta_1) = 1$ 和 $V_{v'}(\alpha, \beta_2) = 1$ 是否同时成立来验证签名 (β_1, v', β_2) 。

② 前述（完整）的签名方案的简单实现中调用了在整个签名过程中（对大量文件）生成，且保存于（安全）存储器上的基本（一次性）签名方案的所有实例。然而，需要指出，有可能重构生成这个实例时所用到的随机值，由伪随机函数就能判定前者（应用与树的顶点相对应的名称）。事实上，这个伪随机函数的种子将是所产生的（完整）签名方案的签名密钥的一部分。

C.7 通用的密码协议

设计实现任意期望功能的安全协议是现代密码学的一个主要部分。反过来看,任何密码方案的设计均可看成是要设计能实现相应功能的安全协议。不过,我们仍然认为有必要区分基本密码学原语(其中几乎不含交互),如加密和签名方案,与通用的密码协议。

本节回顾有关安全多方计算的一般结果,其中两方情形是一种最重要的特例。简言之,这些结果表明可以构造一些协议来安全地计算任意期望的多方函数。实际上,这些协议最引人注目的是其通用性,而我们认为使结论成立的(各种不同)条件并不影响协议的通用性。

构造(m 方)密码协议的通用框架是定义一个随机进程^①,将 m 个输入映射为 m 个输出。进程的输入可被视为 m 个参与方的局部输入, m 个输出是其相应的局部输出。随机进程描述了所需要的功能。即如果 m 个参与方互相信任(或信任外部参与者),则他们可以将局部输入发送给可信方,它将计算出进程的结果并给每个参与方发送相应的输出。在密码协议领域,一个关键问题是由这些互不信任的参与方能将这个(虚构的)可信方“模拟”到什么程度。

本节的综述描述了有可能实现“模拟”的种种模型。这种模拟的差别在于其所依赖的通信信道、限定敌手行为的参数,以及对可信方模拟的层次(即“安全”层次)等假设。我们的处理只涉及独立运行程序的安全性。在一个许多协议都会受到攻击的环境中保持安全性,则超出了本节的讨论范围。感兴趣的读者可参阅参考文献[92]中的7.7.2节。

C.7.1 定义方法及模型

首先进一步讨论“模拟可信方”的概念,这是定义安全多方计算的基础。这种方法遵从仿真范式(见附录C.4.1),如果无论可行的敌手在攻击之后能获得什么,它通过初始的行为也能获得这些信息,则认为一个方案是安全的。在多方计算的通用环境中,将参与真实协议执行过程的敌手所造成的影响,与参与在可信方帮助下计算所要求函数的简单(理想)协议的虚拟执行过程的敌手所造成的影响进行对比。如果无论敌手在真实环境中获得什么,在理想环境中都能获得,那么称真实的协议“模拟了理想环境”(即“模拟了可信方”),而且视其为安全的。这种基本方法适用于在各种不同模型中定义安全性。^②本小节首先讨论用于定义不同模型的参数,并讨论在两种重要情况下的应用。进一步的讨论参见参考文献[92]中的7.2节和7.5.1节。

C.7.1.1 用于定义安全模型的参数

依据确切的(真实的)计算来描述如下参数。在某些情况下,通过在理想模型上施加一

① 也就是说,我们考虑安全地计算随机函数,而不只是安全地计算函数。具体地,考虑任意一个(随机化)进程 F ,对于输入 $(x_1, x_2, \dots, x_m)$,首先随机选择任意一个 m 元函数 f (只依赖于 $l = \sum_{i=1}^m |x_i|$),再输出 m 元组 $f(x_1, x_2, \dots, x_m) = (f_1(x_1, x_2, \dots, x_m), \dots, f_m(x_1, x_2, \dots, x_m))$ 。换言之, $F(x_1, x_2, \dots, x_m) = F'(r, x_1, x_2, \dots, x_m)$,其中 r 在 $\{0, 1\}^{l'}$ 中均匀选择(其中 $l' = \text{poly}(l)$),而是将 $m+1$ 长序列映射为 m 长序列。

② 这里有必要给出一些技术上的说明。首先,假设各方的输入都有相同的长度。只要输入是多项式长度的,通过填充可以执行上述规定。另一方面,一些长度限制对安全结果至关重要,因为一般来说不可能隐藏关于协议输入长度的信息。其次,假设所需的函数在概率多项式时间内是可计算的,因为希望安全协议在概率多项式时间内运行(协议不可能比相应的核心算法更高效)。显然,使用适当的填充,可将结果扩展到在任何给定的(时间构成)时间界限上可计算的函数。

些限制和规定来得到相应的安全性定义。^①在所有情况下定义安全性时，都要求对任意真实情况下的敌手，存在相应的理想模型敌手，两者得到的信息相同。

通信信道

密码学中大部分工作假设通信是同步的，且在任意一对用户间存在点到点的信道（即全连接网络）。进一步地，还假设敌手不能对信道上传递的消息进行篡改（或删除、添加）。在标准模型下，敌手可以窃听所有的通信信道，得到信道上传递的所有信息。另一种模型称为私有信道模型，其中假设敌手得不到诚实参与方之间传递的信息。事实上，在某些情况下，标准模型可以模拟私有信道模型（如利用一个安全的加密方案）。

环境假设

除非有不同的规定，否则，不作环境上的假设（除了明显假设协议各参与方有相同的协议程序副本）。

计算限制

一般关注计算受限的敌手（如概率多项式时间敌手）。然而，私有信道模型允许（有意义地）考虑计算能力无限的敌手。^②

受限的敌手行为

模型的参数包括敌手为“主动的”或“被动的”（即不诚实的参与方是否采取主动的操作来干扰协议的执行，或者仅收集信息），还有敌手是否为“适应性的”（即不诚实的参与方集合是否是在执行前已经确定，还是在执行过程中由敌手适应性地选择）。

受限的安全性

一个重要的例子是愿意容忍“不公平”的协议，其中不诚实的参与方可以（随时）中止协议的执行。要强调在协议执行中止时，不诚实的参与方并不比正常执行获得更多的信息。（诚实的参与方将不会获得所需的输出，但能检测到执行中止）。我们强调，给出这种限制模型的动机是，不可能在无限制模下获取一般的安全两方计算。

不诚实参与方的数量上限

在一些模型中根据需要对此作了假设。例如，在一些模型中，只有当大部分的参与方是诚实的，才有可能实现安全多方计算。

C.7.1.2 举例：多数成员诚实的多方协议

考虑一个主动的、非适应性的、计算受限的敌手，而且不存在私有信道。目标是定义一个在多数成员是诚实的情况下能够保持安全性的多方协议（本小节结尾将讨论需要多数成员诚实的原因）。

首先观察到在任何多方协议中，协议开始执行之前每一方都有可能改变自己的局部输入，即使是引入一个可信方也不能避免。结果，在真正执行时敌手这样的行为（即在协议执行前修改自己的输入）不认为会影响安全性。一般来说，人们认为即使有可信方参与时也无法避免的事件不会影响安全性。我们希望安全协议（在真实模型中）仅受到那些存在可信第三方时也不能避免的因素影响。于是，安全多方计算定义所依据的基本范式，可归结为要求安全

① 例如，在两方计算（见附录 C.7.1.3）中，只有当提前终止不被看作是破坏安全的行为时，安全计算才有可能。此时，适当的安全定义（由仿真范式）必须允许在理想模型下，任意一方能提前中止。

② 我们强调，对于计算能力无限的敌手，在定义安全性时，也要求对每个现实敌手，无论敌手在参与实际协议执行之后，可以计算出何种结果，该结果都可由想象中的敌手在参与虚拟的平凡理想协议（在可信方的帮助下用于计算期望功能）之后，在相当的时间内求得。也就是说，虽然在现实模型下对敌手没有做出任何计算上的限制，但要求理想模型下的敌手可在相当时间内得到同样的效果（即是要模仿的现实模型敌手运行时间的多项式）。

协议在实际执行时的情况，等同于协议在相应的理想模型中（其中参与方可引入可信方）执行时的情况。换句话说，在安全协议中参与方的“有效故障”被限制为在相应理想模型中的公设。

当定义安全多方协议（多数成员诚实）时，要指出在理想模型下（即引入可信方时）什么是不能避免的。由于我们感兴趣的是多数成员诚实时协议的执行，所以考虑只有少数成员能按如下方法合谋的理想模型：

（1）少数不诚实成员共享其原始输入，并替换要发送给可信方的输入（其他成员分别将其原始输入发送给可信方）。

（2）收到所有成员的输入后，可信方判定相应的输出并发送给相应的参与方（要强调，少数不诚实的合谋者看不到在诚实成员和可信方之间传输的信息）。

（3）从可信方收到输出消息后，每个诚实方各自产生自己的输出，而少数不诚实的合谋者基于他们所了解的信息（即他们的初始输入和所得到的输出）来确定其输出。

多数成员诚实的安全多方计算需要模拟这个理想模型。也就是说，在协议执行中，任何可行的敌手想去控制大多数参与方的努力，都可以通过一个在理想模型中（不同的）可行的敌手控制相应的参与方来模仿。

定义 C.17（安全协议—概述） 令 f 是 m -进制的函数， Π 是运行在真实模型中的 m -方协议。

- 对于控制了少量参与方（且攻击所有通信信道）的真实模型敌手 A 和一个 m -序列 $\bar{x}$ ，用 $\text{REAL}_{\Pi,A}(\bar{x})$ 表示在敌手 A 的攻击下，从 Π 对输入 $\bar{x}$ 的执行中得到的 m 序列。

- 对于控制了少量的参与方理想模型的敌手 A' 和一个 m -序列 $\bar{x}$ ，用 $\text{REAL}_{f,A'}(\bar{x})$ 表示在敌手 A' 的攻击下，从前述的三步理想进程对输入 $\bar{x}$ 的执行中得到的 m 序列。

如果对所有控制了多数成员的可行真实模型敌手 A ，存在一个控制了相同成员的可行理想模型敌手 A' ，使得概率空间 $\{\text{REAL}_{\Pi,A}(\bar{x})\}_{\bar{x}}$ 和 $\{\text{REAL}_{f,A'}(\bar{x})\}_{\bar{x}}$ 是计算上不可区分的（如定义 C.5），则称 Π 安全地实现了多数成员诚实的 f 。

于是，安全性意味着在安全协议的真正执行中每个少数群体的行为，“从本质上被约束”到在协议开始之前只能替代自己的局部输入（独立于少数参与者的局部输入），在协议终止后只能替代它自己的局部输出（仅依赖于它自己的输入和输出）（要强调在真实执行中，少数参与方能获得额外的信息；由于他们自己可以复制这些信息，因此在安全协议中要求他们从这些附加信息仍然得不到任何东西）。

定义 C.17 没有涉及到私钥信道模型，体现在真实模型的定义中允许敌手窃听所有的信道，从而可影响概率空间 $\{\text{REAL}_{\Pi,A}(\bar{x})\}_{\bar{x}}$ 。在私有信道模型下定义安全性时不允许真实模型敌手攻击诚实参与方之间的信道，这也会影响概率空间 $\{\text{REAL}_{\Pi,A}(\bar{x})\}_{\bar{x}}$ 。另一方面，当我们希望定义针对被动敌手的安全性时，真实模型敌手的范围和理想模型敌手的范围都会改变。在真实模型执行中，所有参与方遵守协议，但敌手可能会依据自己的内部状态（在整协议执行期间）来随意影响输出。在相应的理想模型中，不允许敌手修改诚实参与方的输入（在第（1）步），但允许修改他们的输出（在第（3）步）。

在不诚实参与方占据多数的情况下，也能给出类似于定义 C.17 的定义。这样的定义看起来更自然，但问题是这样一个定义无法满足。也就是说，在至少一半参与方不诚实且使用了适当的攻击策略时，对于大部分（自然的）函数而言，不存在计算它们的安全多方协议。这

可从对两方计算的一个不可能的结果看出来，它断言没有办法阻止一个参与方提前挂起协议的执行。另一方面，如果协议执行的提前终止不被视为违反安全性，则多数成员不诚实的安全多方计算是有可能的（见 C.7.1.3）。

C.7.1.3 另一个例子：允许中止的两方协议

根据上面最后的讨论，现在考虑多方计算，其中提前终止执行不认为是违背了安全性。为简化起见，考虑两方计算的特例（如附录 C.7.1.2，考虑非适应性的、主动的、计算上有界的敌手）。

直观上，在任何两方协议中，每一方可以在任何时间点挂起，而且一旦得想要的输出随时可以这样做。于是，在一方的输出需要依赖双方的输入时，一个参与方已获得想要的输出而阻止另一方完全地决定自己的输出，这种情况总有可能发生。^①即使在两方仅希望生成一个公共随机值时这种现象也会发生。于是，当考虑两方场合下的主动攻击者时，不认为这种对执行的提前挂起会违背安全性。因此，考虑理想模型，其中两个参与方可能在任何时间点关掉可信（第三）方。特别是，在可信方将计算结果给了一个参与方，而在把计算结果给另一方之前时可能会这样做。于是，在相应理想模型中的执行过程如下：

（1）每一方将自己的输入发送给可信方，其中不诚实的参与方可能会替换自己的输入，或者根本不发送（可以通过发送一个默认值来欺骗）。

（2）在收到两个参与方的输入后，可信方确定相应的一对输出，将第一个输出发送给第一个参与方。

（3）如果第一个参与方是不诚实的，它可能会将可信方引向停止，否则，它会一直引导可信方运行。如果被引导到运行，可信方将第二个输出发送给第二个参与方。

（4）在从可信方得到输出消息后，诚实的参与方在本地产生输出，而不诚实的参与方可能会基于所有已知信息（即它的初始输入和它收到的输出）来确定自己的输出。

允许中止的安全两方计算需要仿真这个理想模型。也就是说，如定义 C.17，在定义安全性时，要求对每个可行的真实模型敌手 A ，存在一个控制了相同参与方的可行的理想模型敌手 A' ，使得表示相应（真实的和理想的）执行的概率空间在计算上是不可区分的。这意味着，在安全协议中每一方的“有效故障”被约束为按其意愿提供初始输入，并在任意时间点中止计算（不用说，每一方初始输入的选择不必依赖于其他参与者的输入。）

处理提前挂起执行问题（即中止）的另一种方式是，将注意力集中到单输出函数上，即该函数中只有一个参与方被挂起来以得到输出。除对不诚实参与的集合没有约束外（特别是在两方计算中可考虑一个不诚实参与方），安全函数的安全计算的定义可以和定义 C.17 一样。进一步的讨论见参考文献[92]中的 7.2.3 节。

C.7.2 一些已知结论

本小节列举已知的通用安全多方计算的模型（即用于构造可以计算任何期望的函数的安全多方协议的模型）。第一个结论由 Goldreich、Micali、Wigderson 和 Yao 给出^[100, 101, 241]。

在标准信道模型中

假定存在增强的陷门置换，^②安全多方计算在以下 3 种模型下是可能的（参见参考文献[100, 101, 241]，细节见参考文献[92]中的第 7 章。

① 相比之下，在多数成员诚实的情况下（参考附录 C.7.1.2），诚实的参与方没有得到自己的输出的情况并不少见。它可以向其他的诚实参与方寻求帮助，他们联合在一起形成多数，可以做到少数不诚实者做不到的事，见附录 C.7.3.2。

② 见脚注。

- (1) 对任意数量的不诚实参与方的被动敌手（见参考文献[92]中的 7.3 节）。
- (2) 仅可控制少数参与者主动敌手（见参考文献[92]中的 7.5.4 节）。
- (3) 对任意数量的不诚实参与方的主动敌手，倘若执行的挂起（如附录 C.7.1.3 中讨论的）不被视为违反安全性（见参考文献[92]中的 7.4 节和 7.5.5 节）。

在所有情况下，敌手都是计算受限的，且为非适应性的。另一方面，敌手可能会攻击各诚实方之间的通信线路（即这里不假设“私有信道”）。对主动攻击者的结果假设需要广播信道。事实上，用一个签名方案，且假设每一方知道（或能可靠地得到）对应于每个其他参与方的验证密钥，就可以实现广播信道。

在私有信道模型中

无计算上的假设，且允许计算能力无限的敌手，但是假设私有信道存在，则在以下两种模型下存在安全多方计算（见参考文献[34, 53]）：

- (1) 只能控制少数参与方的被动敌手。
- (2) 只能控制少于 $1/3$ 参与方的主动敌手。

两种情况下敌手都可以是适应性的。

C.7.3 构造方法及两个简单协议

本小节简要描述用于构造安全多方协议的一些范式。主要讨论计算受限且非适应性的敌手模型下安全协议的构造^[100, 101, 241]。构造过程分为两步：首先，给出一个被动敌手（任意数量不诚实参与方的）模型，然后将这个协议“编译”为在两个主动敌手模型之一（即或者仅允许敌手控制少量参与方，或者提前挂起不被视为违反安全性的模型）中安全的协议。这两个步骤分别见于以下两个小节，其中针对两种特定任务分别给出两个相对简单的协议，它们可以扩展为通用协议。

回顾在被动敌手模型中，所有的参与方遵守既定的协议，但在终止时敌手可能会依据其中间内部状态替换不诚实参与方的输出（在整个执行过程中）。后面将用术语被动安全（主动安全）表示在被动（主动）敌手模型下安全的协议。

C.7.3.1 带有共享份额的被动安全计算

为简单起见，只考虑确定型 m 方计算的特例。假设各参与方持有用于计算对任意长度输入的函数值的电路，该电路仅包括“与”门和“非”门。关键的想法是让参与方与其他人“秘密地共享”其输入，并让参与方将电路输入线的份额“秘密地转换”为电路输出线的份额，进而得到输出份额（允许重构出真正的输出）。电路中每条线上的值都被共享，使得由所有的份额可以产生一个值，而即使缺少一个份额也会这个值完全无法被确定。换言之，使用一个简单的秘密共享方案，使得一个比特位 b 被一个 m 比特的随机序列共享，该序列的所有位之和 mod 2 等于 b 。首先，每个参与方与所有参与者共享自己的输入比特（通过给每个参与者秘密发送一个随机值，并相应地设置自己的份额）。然后，所有参与方联合扫描电路，从自己的输入线到自己的输出线，对每个门的处理方式如下：

- 当遇到一个门时，参与方已经持有关于门的输入线上值的共享份额，其目标是得到关于门的输出线上的值的共享份额。
- 对于非门是很容易的：第一个参与方仅将其份额值取反即可，其他参与方保留自己的份额。

• 由于与门与模 2 乘相对应,参与方需要安全地计算如下的随机函数(其中用 x_i 表示一条输入线上的份额,用 y_i 表示第二第输入线,用 z_i 表示输出线上的份额,用序号 i 来表示参与方 i 所持有的份额),即

$$((x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)) \mapsto (z_1, z_2, \dots, z_m) \quad (C.1)$$

其中

$$\sum_{i=1}^m z_i = \left(\sum_{i=1}^m x_i \right) \cdot \left(\sum_{i=1}^m y_i \right) \quad (C.2)$$

即 z_i 由式(C.2)随机决定。

最后,参与方将每条电路输出线上自己的份额发送给目标参与方,目标参与方重构相应的比特值。于是,参与方通过重复计算式(C.1)和式(C.2)的 m 进制函数的被动安全计算,将电路输入线上的份额转换为电路输出线上的份额。也就是说,将安全计算整个(任意)电路“约减”为安全地计算特定(很小的)多方计算。但实际上更简单:关键之处在于

$$\left(\sum_{i=1}^m x_i \right) \cdot \left(\sum_{i=1}^m y_i \right) = \sum_{i=1}^m x_i y_i + \sum_{1 \leq i < j \leq m} (x_i y_i + x_j y_j) \quad (C.3)$$

于是,可按如下步骤计算式(C.1)和式(C.2)的 m 进制函数(其中所有的算术操作都是模 2 的):

(1) 每个参与方 i 在本地计算 $z_{i,i} \stackrel{\text{def}}{=} x_i y_i$ 。

(2) 然后,每个参与方(即参与方 i 和 j)安全地计算 $x_i y_j + y_i x_j$ 的随机份额。即参与方 i 和 j (分别持有 (x_i, y_i) 和 (x_j, y_j)),要安全地计算随机的两方函数 $((x_i, y_i), (x_j, y_j)) \mapsto (z_{i,j}, z_{j,i})$,其中 z 随机地等于 $z_{i,j} + z_{j,i} = x_i y_j + y_i x_j$ 。等价地,参与方 j 均匀选择 $z_{j,i} \in \{0, 1\}$,参与方 i 和 j 安全计算如下的确定性函数,即

$$((x_i, y_i), (x_j, y_j, z_{j,i})) \mapsto (z_{j,i} + x_i y_j + y_i x_j, \lambda) \quad (C.4)$$

式中: λ 表示空字符串。

(3) 最后,对每个 $i=1, 2, \dots, m$, 由和 $\sum_{j=1}^m z_{i,j}$ 得到参与方 i 想要的份额。

上述的构造方法是对参考文献[101]中给出的构造方法的模仿。详细的描述和完整证明可参见参考文献^[92]中的 7.3.4 节和 7.5.2 节。

上述构造将任意 m 进制函数的被动安全计算约减为式(C.4)的简单二进制函数。后者通过被动安全的方式以 1-out-of-4 的不经意传输实现。粗略地说, 1-out-of- k 不经意传输是使一个参与方能够知道另一个参与方持有 k 个秘密之一的协议,而第二个参与方不知道哪个秘密被第一个参与方得到。也就是说,它允许式(C.5)两方函数的被动安全计算,即

$$(i(s_1, s_2, \dots, s_k)) \mapsto (s_i, \lambda) \quad (C.5)$$

注意: 任意函数 $f: [k] \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{\lambda\}$ 都可以通过调用对输入 i 和 $(f(1, y), (f(2, y), \dots, f(k, y)))$ 的 1-out-of- k 不经意传输,以被动安全的方式计算,其中 $i(y)$ 是第一个(第二个)参与方的初始输入。

被动安全的 1-out-of- k 不经意传输

当限制秘密为单比特时,用一组改进的陷门置换 $\{f_a: D_a \rightarrow D_a\}_{a \in \bar{I}}$ 和相应的核心断言 b , 可以实现式(C.5)中函数的被动安全计算。

输入: 第一个参与方称为接收方,输入 $i \in \{1, 2, \dots, k\}$ 。第二个参与方称为发送方,输入 $(\sigma_1,$

$\sigma_2, \dots, \sigma_k) \in \{0, 1\}^k$ 。

步骤 S1: 发送方随机选择一个带有陷门 t 的置换 f_α , 并将置换 f_α (即序号 α) 发送给接收方。

步骤 R1: 接收方均匀且独立地选择 $x_1, x_2, \dots, x_k \in D_\alpha$, 对所有 $j \neq i$, 令 $y_i = f_\alpha(x_i)$ 和 $y_j = x_j$, 并将 $(y_1, y_2, \dots, y_k)$ 双发送给发送方。

于是, 接收方知道了 $f_\alpha^{-1}(y_i) = x_i$, 但对任意 $j \neq i$ 无法判断 $b(f_\alpha^{-1}(y_j))$ 。不用多说, 后一个断言假定接收者执行协议 (即仅考虑被动攻击)。

步骤 S2: 当发送接收到 $(y_1, y_2, \dots, y_k)$ 后, 用陷门求逆算法和陷门 t , 发送方对每个 $j \in \{1, 2, \dots, k\}$ 计算 $z_j = f_\alpha^{-1}(y_j)$ 。将 k 元组 $(\sigma_1 \oplus b(z_1), \sigma_2 \oplus b(z_2), \dots, \sigma_k \oplus b(z_k))$ 发送给接收者。

步骤 R2: 接收到 $(c_1, c_2, \dots, c_k)$ 后, 接收方在本地输出 $c_i \oplus b(x_i)$ 。

首先, 观察到这个协议正确地执行了 1-out-of- k 不经意传输, 即接收方的本地输出 (即 $c_i \oplus b(x_i)$) 实际上等于 $(\sigma_i \oplus b(f_\alpha^{-1}(f_\alpha(x_i)))) \oplus b(x_i) = \sigma_i$ 。然后, 给出直观的分析, 表明为什么这个协议安全实现了 1-out-of- k 不经意传输。直观地, 发送方没有从执行中获得任何信息, 原因是对 i 的可能值, 发送方观察到了相同的分布; 确切地说, 一系列均匀的 k 和 D_α 的独立分布的元素 (实际上, 关键在于对 D_α 上独立分布的元素应用 f_α 会产生一个 D_α 上均匀分布的元素)。对于接收方, 它没有从执行中得到计算上的知识, 原因是对 $j \neq i$, 接收方能看到在关于 σ_j 的唯一信息是三元组 $(\alpha, x_j, \sigma_j \oplus b(f_\alpha^{-1}(x_j)))$, 其中 x_j 在 D_α 上均匀分布, 而由此信息去预测 σ_j 是不可行的, 其难度与随机猜测相同^① (详细的证明见参考文献[92]中的 7.3.2 节)。

C.7.3.2 从被动安全协议到主动安全协议

如何将任意被动安全协议转换为相应的主动安全协议。两个协议使用的通信模型都包括一个广播信道。注意: 可以假设原始协议中的消息是在广播信道上传输的, 因为敌手肯定可以得到这些消息 (通过在点对点信道上进行窃听), 且在被动敌手情况下广播信道是易实现的。对于得到的主动安全协议, 通过一个 (认证的) Byzantine 协商协议, 可以实现所使用的广播信道, 于是在标准点对点模型上 (其中不存在广播信道) 给出对这个模型的模拟。认证的 Byzantine 协调一般用签名来实现 (且每一方均知道对应于其他每个参与者的验证密钥)。

再看传输过程, 主要思想是用零知识证明 (附录 C.4.3.3 中所描述的) 以迫使参与方的行为与 (被动安全) 协议的方式一致。确切地说, 需要将每一方限制为唯一的一致行为 (即依据固定的本地输入和一系列随机掷币的值), 且要保证每个参与方不能依据不诚实参与方的输入 (且/或自己的掷币值) 来固定自己的输入 (且/或自己的掷币值)。于是, 在对原始协议的一步模拟开始之前, 需要一些预备步骤。确切地说, 演变出的协议 (类似于原始协议, 在广播信道上运行) 运行如下:

(1) 承诺本地输入。在模拟原始协议之前, 每一方承诺自己的输入 (使用附录 C.4.3.1 中定义的承诺方案)。而且, 每一方还要用零知识证明 (见 9.2.3 节) 来证明他知道自己输入, 即证明他能够对所发出的承诺进行解承诺 (这些零知识证明能防止不诚实的参与方依据不诚实参与方的输入来设置他们的输入)。

(2) 本地随机记录的生成。所有参与方共同为每个参与方生成一系列随机比特, 使得只有这个参与方知道所生成的随机序列的结果, 每个人都可以得到关于这个结果的承诺。这些

① 后一种直观分析中假设抽样 D_α 是平凡的 (即在抽样用到的随机掷币算法和所得到的抽样之间存在易于计算的关联性), 而一般结果中算出用于抽样的掷币算法。这就是在上述协议的一般分析中用改进的困难假设的原因。

序列将被当作原始协议的随机输入（即掷币序列）。为参与方 X 生成的随机序列的每一比特位，由参与方 X 和其他每个参与方之间进行的（扩展的）掷币协议（见参考文献[92]中的 7.4.3.5 节）的输出结果异或产生。后面的协议为其他参与方提供了对参与方 X 所得到的结果的承诺。

（3）提前终止的有效防止。另外，当要演变成（从被安全协议到主动安全协议）允许对手仅控制少数参与方的模型时，每一方通过一个“可验证秘密共享”（VSS）协议（见参考文献[92]中的 7.5.5.1 节）共享他的输入和随机输入。粗略地讲，VSS 协议允许每个参与方验证所得到的份额适合公开发布的消息，包括对所有份额的承诺，其中通过一定数量的份额可以恢复出秘密，VSS 以这种方式允许共享秘密份额。VSS 的使用确保如果参与方 X 提前挂起执行，则诚实的参与方能够一起重构出参与方 X 的所有秘密，并以他的角色继续协议的执行。这一步有效地阻止了提前终止，而且在将提前终止不被视为违反安全性的模型中也不是必须的。

（4）原始协议的逐步模仿。一旦上述步骤全部完成，新协议就模仿了原始协议的每一步。在每一步，下一条消息（如原始协议中所确定的）是发送方自己输入、随机输入，以及截止目前所收到的消息（由于是通过广播信道发送的，所以所有人都知道）的一个函数。而且，发送方的输入由它的承诺（如步骤（1）中所发送的）和用同样方法确定的（在步骤（2））随机输入来决定。于是，下一条消息（如原始协议中所确定的）是公开的已知字符串（即所谓的承诺和通过广播信道发送的其他消息）的一个函数。而且，下一条消息实际是正确计算的断言是一个 NP-断言，并且发送方知道相应的 NP-证据（即自己的输入和随机输入以及相应的解承诺信息）。于是，发送方可以通过零知识的方式（向其他每个参与方）证明发送的消息实际上是根据原始协议计算得到的。

以上的演变首先在参考文献[100, 101]中给出。详细描述和完整证明在参考文献[92]中的 7.4 节和节 7.5 节中给出。

安全掷币协议。使用承诺方案，描述一个安全的（普通的而不是增强的）掷币协议。

步骤 C1: 参与方 1 均匀选择 $\sigma \in \{0,1\}$ ，并将对 σ 的承诺 c 发送给参与方 2。

步骤 C2: 参与方 2 均匀选择 $\sigma' \in \{0,1\}$ ，并将 σ' 发送给参与方 1。

步骤 C3: 参与方 1 输出值 $\sigma \oplus \sigma'$ ，并将 σ 和解承诺信息 d 一起发送给参与方 2。

步骤 C4: 参与方 2 检查 (σ, d) 是否符合在步骤 C1 中获得的承诺 c 。如果满足条件，则输出 $\sigma \oplus \sigma'$ ，否则，终止并输出 $\perp$ ，其中 $\perp$ 表示参与方 1 有效地提前中止了协议。

输出：参与方 1 总是输出 $b = (\text{def}) \sigma \oplus \sigma'$ ，而参与方 2 输出 b 或 $\perp$ 。

步骤 C1 和步骤 C2 可直观地视为“掷币到满意”。此时（即在步骤（2）之后），币的值就被确定（本质上相当于一个随机值），但仅有一个参与方（即参与方 1）“能看到”（即知道）这个值。显然，如果两个参与方都是诚实的，则他们会输出相同的、分别从步骤 C3 和步骤 C4 恢复出来的随机值。直观地，每一方能确保输出是均匀分布的，参与方 1 通过错误地执行步骤（3）可以引导协议的提前终止。正式地说，必须要表示如何通过一个合适的理想模型（其中提前终止是被允许的），来仿真任意实际模型的对手的影响。参考文献[92]中的 7.4.3.1 节完成了这个工作。

C.7.4 结语

附录 C.7.1 和附录 C.7.2 给出大量与安全多方计算相关的定义和结果，其中有些定义与其他定义不可比较（既不蕴含其他定义，也不被其他定义所蕴含）。例如，在附录 C.7.1.2 和附录 C.7.1.3 中，给出了“安全多方计算”两种定义，一个需要大多数人诚实，而另一个则允许

中止。这些定义是不可比较的，没有通用的理由更倾向于某一个。确切地说，如附录 C.7.1.2 所述，可以给出一个蕴含二者的自然的定义（即放弃定义 C.17 中所诚实参与方的数量界限）。的确，所得到的定义没有以前两个定义中那些复杂的约束条件。新定义中“仅有”的问题是，它无法被满足（通常）。因此，在这个附录中第一次碰到了自然的（一般的）定义无法满足的条件，我们只能从两个较弱的替代方案中选择，每个替代方案都带有本质的缺陷。

通常，附录 C.7 比前面的小节具有很强的折中味道（即承认先天的限制，满足受限制的有意义的目标）。相比之下，本附录中其他部分给人的印象，现在很明显不能得到所有我们想要的（这没有涉及到在并发组合中保持安全性的问题，见参考文献[92]中的 7.7.2 节）。相反，我们应该去研究替代方案，得到最适合实际需要的方案。

实际上，如附录 C.1 所述，事实上，我们能够定义密码学目标，并不意味着就能按所定义的满足它。当不能满足初始的定义时，应该去寻找弱化和能被满足的条件。这种条件的弱化应该有清晰的定义，明确什么是能达到的（和不能达到的）。如此，可以允许弱化后的条件应用于特定场合。这似乎可以作为一个很好的观点来结束本附录。

所有人的最重要的利益能得到满足的方案就是好的折中方案。

Adv. Klara Goldreich–Ingwer (1912—2004)

附录 D

概率论基础及随机性中的前沿问题

这是什么？咖喱鸡肉和海鲜沙拉？很好，但是放在同一个盘子里？让人恶心！约翰·哈斯塔德 于格伦德尔，剑桥（1985）

内容提要

本附录整理了一些有关概率论的预备知识和相关的前沿性内容，这些知识与随机性理论在计算中的用途相关。每个专题都用一个单独的小节阐述。

概率论预备知识除了包括贯穿全书的随机变量相关知识外，还包括 3 个有用的不等式：马尔可夫不等式、切比雪夫不等式、切诺夫界。

前沿性内容包括散列、抽样和随机性抽取。对于散列，我们描述两两（或 t -元）独立 Hash 函数对的构造，以及剩余散列引理(Leftover Hash Lemma)及其变形。然后回顾“抽样复杂性”，即评估定义域较大的任意函数的平均值时，涉及到的样本数量和随机复杂性。最后给出从较弱（或有缺陷）随机源中提取几乎完美随机数的方法综述。

D.1 概率论基础

概率在复杂性理论中扮演着主要角色（如在 6.9 节中可以看到）。假设本书读者已经熟悉基础性的概率论知识，因此在本节中只介绍整本书使用的概率符号和 3 个有用的概率不等式。

D.1.1 符号约定

本书只涉及离散的概率分布。具体地，基本概率空间由均匀选择的长为 l 的字符串集构成。即样本空间是所有 l 比特串的集合，每个串被分配的概率测度为 2^{-l} 。一般来说，随机变量被定义为从样本空间到真实空间的函数。当函数从样本空间映射到一组二进制字符串时，也使用随机变量这个词。另一方面，常常不指定概率空间，而是直接讨论随机变量。例如，或许会说 X 是一个集合中的所有字符串的随机变量赋值，这样 $\Pr[X=00]=1/4$ ， $\Pr[X=111]=3/4$ （这样一个随机变量可以定义在样本空间为 $\{0, 1\}^2$ 上，因此， $X(11)=00$ ， $X(00)=X(01)=X(10)=111$ ）。随机变量的一个重要实例是随机过程（例如，在 6.1 节中的概率多项式时间算法）的输出。

所有的概率断言针对着预先定义的随机变量（随机变量的函数）。通常可令 $\Pr[f(X)=1]$ ，

这里 X 是一个预先定义的随机变量, 而 f 是一个函数。一个重要的习惯是, 在一个概率断言内出现的所有相同的符号, 都指代相同的 (独特的) 随机变量。因此, 如果 $B(\cdot, \cdot)$ 是关于两个变量的布尔表达式, X 是一个随机变量, 那么, $\Pr[B(X, X)]$ 代表以 $\Pr[X=x]$ 概率挑选 x 时 $B(x, x)$ 的概率。例如, 对任意一个随机变量 X , 有 $\Pr[X=X]=1$ 。我们强调, 如果要讨论 x 和 y 以相同的概率分布独立地选择 $B(x, y)$ 的概率, 这样将定义具有相同的概率分布, 两个独立的随机变量。于是, 如果 X 和 Y 是两个独立的随机变量, 那么, $\Pr[B(X, Y)]$ 代表以 $\Pr[X=x] \cdot \Pr[Y=y]$ 概率选择 (x, y) 时 $B(x, y)$ 的概率。例如, 对于任意两个独立的随机变量 X 和 Y , $\Pr[X=Y]=1$, 当且仅当 X 和 Y 均为平凡的 (即将所有概率质量赋予单个字符串)。

在本书中, U_n 表示在长度为 n 的字符串集合中均匀分布的随机变量。即当 $\alpha \in \{0, 1\}^n$ 时, $\Pr[U_n = \alpha] = 2^{-n}$, 否则为 0。通常定义 U_n 分布为均匀分布 (忽略其为在 $\{0, 1\}^n$ 上的均匀分布)。

另外, 将会使用在 $\{0, 1\}^n$ 或者 $\{0, 1\}^{(\ell(n))}$ 上任意分布的随机变量, $\ell: \mathbb{N} \rightarrow \mathbb{N}$ 为某个函数。这样一些随机变量通常可以用 X_n 、 Y_n 、 Z_n 等符号来表示。在这里我们要着重强调的是, 在某些情况下, X_n 通常为 $\{0, 1\}^n$ 上的分布, 因为在其他情况下它为 $\{0, 1\}^{(\ell(n))}$ 上的分布, 而 $\ell(\cdot)$ 通常为一个多项式函数。我们还将常常提及概率空间, 这是随机变量 $\{X_n\}_{n \in \mathbb{N}}$ 的无限序列, 每一个 X_n 的取值范围为字符串长度 n 的有限多项式。

统计差异

两个随机变量 X 和 Y 的统计距离 (或变化距离) 定义如下:

$$\frac{1}{2} \cdot \sum_v |\Pr[X=v] - \Pr[Y=v]| = \max_S \{\Pr[X \in S] - \Pr[Y \in S]\} \quad (\text{D.1})$$

如果 X 和 Y 的统计距离最大 (或最小) 为 δ , 那么, X 距离 Y 是 δ -接近 (或 δ -远离) 的。

D.1.2 3 个不等式

以下 3 个概率不等式非常有用。这 3 个不等式涉及到被赋予真实值的随机变量, 而且还给出了当其偏离期望时的概率上限。

D.1.2.1 马尔可夫不等式

马尔可夫不等式是最基本的不等式, 对于任意随机变量规定了最大值和最小值。简言之, 马尔可夫不等式说明了随机变量的下界为零, 可以表述如下: 令 X 为非负的随机变量, v 为非负实数, 有

$$\Pr[X \geq v] \leq \frac{E(X)}{v} \quad (\text{D.2})$$

等价地, $\Pr[X \geq r \cdot E(X)] \leq \frac{1}{r}$ 。证明如下:

$$\begin{aligned} E(X) &= \sum_x \Pr[X=x] \cdot x \geq \sum_{x < v} \Pr[X=x] \cdot 0 + \sum_{x \geq v} \Pr[X=x] \cdot v \\ &= \Pr[X \geq v] \cdot v \end{aligned}$$

D.1.2.2 切比雪夫不等式

根据马尔可夫不等式, 当一个随机变量偏离期望时可以得到更强的界, 这个界就叫做切比雪夫不等式, 在得知随机变量一些额外信息时它是很有用的 (特别是对于方差有一个好的上界

时)。对于一个期望值有限的随机变量 X , 并且 $\text{Var}(X)=E(X^2)-E(X)^2$, 切比雪夫不等式如下:

令 X 为一个随机变量, 并且 $\delta > 0$, 则有

$$\Pr[|X - E(X)| \geq \delta] \leq \frac{\text{Var}(X)}{\delta^2} \quad (\text{D.3})$$

证明: 定义一个随机变量 $Y \stackrel{\text{def}}{=} (X - E(X))^2$, 并且使用马尔可夫不等式可得到

$$\Pr[|X - E(X)| \geq \delta] = \Pr[(X - E(X))^2 \geq \delta^2] \leq \frac{E[(X - E(X))^2]}{\delta^2}$$

其推论如下。

推论 (成对独立取样): 切比雪夫不等式在分析通过重复抽样求近似过程的错误概率时起到很大的作用。它假设样本是以独立成对的方式抽取的, 这里 $X_1, X_2, \dots, X_n$ 是成对独立的, 当对任意的 $i \neq j$ 和 α, β , 有 $\Pr[X_i = \alpha \wedge X_j = \beta] = \Pr[X_i = \alpha] \cdot \Pr[X_j = \beta]$ 。推论表述如下:

令 $X_1, X_2, \dots, X_n$ 为成对独立的随机变量, 有相同的期望 μ 和相同的方差 σ^2 , 则对任意一个 $\varepsilon > 0$, 有

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| \geq \varepsilon\right] \leq \frac{\sigma^2}{\varepsilon^2 n} \quad (\text{D.4})$$

证明: 定义随机变量 $\bar{X}_i \stackrel{\text{def}}{=} X_i - E(X_i)$, 这里 $\bar{X}_i$ 都是成对独立且期望为 0。对随机变量 $\sum_{i=1}^n \frac{X_i}{n}$

运用切比雪夫不等式和期望线性算子, 可以得到

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| \geq \varepsilon\right] \leq \frac{\text{Var}\left[\sum_{i=1}^n \frac{X_i}{n}\right]}{\varepsilon^2} = \frac{E\left[\left(\sum_{i=1}^n \bar{X}_i\right)^2\right]}{\varepsilon^2 \cdot n^2}$$

现在再利用期望的线性特性, 可以得到

$$E\left[\left(\sum_{i=1}^n \bar{X}_i\right)^2\right] = \sum_{i=1}^n E[\bar{X}_i^2] + \sum_{1 \leq i \neq j \leq n} E[\bar{X}_i \bar{X}_j]$$

另外, 因为 $\bar{X}_i$ 是成对独立的, 则 $E[\bar{X}_i \bar{X}_j] = E[\bar{X}_i] \cdot E[\bar{X}_j]$, 并且 $E[\bar{X}_i] = 0$, 所以有

$$E\left[\left(\sum_{i=1}^n \bar{X}_i\right)^2\right] = n \cdot \sigma^2$$

接下来就是必然的结果。

D.1.2.3 切诺夫界

当使用成对独立样本点时, 其近似错误概率随着样本点数量的变化而线性减少 (由式 (D.4) 可知); 而当使用完全独立的样本点, 其近似错误概率随着样本点数量的变化而指数减少 (称随机变量 $X_1, X_2, \dots, X_n$ 是完全独立的, 如果对于任意序列 $a_1, a_2, \dots, a_n$ 有

$\Pr\left[\sum_{i=1}^n X_i = a_i\right] = \prod_{i=1}^n \Pr[X_i = a_i]$ 。以下将给出支持上述断言的概率界限。第一个界限，一般就是 0-1 随机变量（随机变量取值为 0 和 1）的切诺夫界：令 $p \leq \frac{1}{2}$, $X_1, X_2, \dots, X_n$ 为独立的 0-1 随机变量，这样对于每个 i 有 $\Pr[X_i = 1] = p$ 。因此，对于任意 $\varepsilon \in (0, p(1-p))$ ，有

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n} - p\right| > \varepsilon\right] < 2 \cdot e^{-\frac{\varepsilon^2}{2p(1-p)}n} \leq 2 \cdot e^{-2\varepsilon^2 n} \quad (\text{D.5})$$

证明概要：上界 $\Pr\left[\sum_{i=1}^n X_i - pn > \varepsilon n\right]$, $\Pr\left[pn - \sum_{i=1}^n X_i > \varepsilon n\right]$ 同样有界。令 $\bar{X}_i \stackrel{\text{def}}{=} X_i - E(X_i)$ ，对随机变量 $e^{\lambda \sum_{i=1}^n \bar{X}_i}$ 使用马尔可夫不等式，这里 $\lambda > 0$ ($\lambda = \theta(\varepsilon/p(1-p))$)，这样可以优化我们

得到的表达式。因此， $\Pr\left[\sum_{i=1}^n \bar{X}_i > \varepsilon n\right]$ 的上界为 $\frac{E\left[e^{\lambda \sum_{i=1}^n \bar{X}_i}\right]}{e^{\lambda \varepsilon n}} = e^{-\lambda \varepsilon n} \cdot \prod_{i=1}^n E\left[e^{\lambda \bar{X}_i}\right]$ ，这里取等号是

因为随机变量的独立性。为了简化余下的证明，我们建立了一个次优的范围。对 e^x 使用泰勒展开（当 $x \leq 1$ 时， $e^x < 1 + x + x^2$ ）并且得到 $E(\bar{X}_i) = 0$, $E\left[e^{\lambda \bar{X}_i}\right] < 1 + \lambda^2 E\left[\bar{X}_i^2\right] = 1 + \lambda^2 p(1-p)$ 。

因此， $e^{-\lambda \varepsilon n} \cdot (1 + \lambda^2 p(1-p))^n < \exp(-\lambda \varepsilon n + \lambda^2 p(1-p)n)$ 的上界为 $\Pr\left[\sum_{i=1}^n X_i - pn > \varepsilon n\right]$ ，当

$\lambda = \varepsilon / (2p(1-p))$ 时最优取得指数 $-\frac{\varepsilon^2}{4p(1-p)} \cdot n$ 。当然，这个方法还可以利用在更一般的条件下，如 $X_i \in [0, 1]$ 而不是 $X_i \in \{0, 1\}$ 。

上述证明方法可应用于更普遍的场合。^①以下给出一个更一般的界限，其中涉及到相互独立的随机变量，每个均有界，但不一定相同（通常称为 Hoeffding 不等式）。令 $X_1, X_2, \dots, X_n$ 为 n 个相同分布的独立随机变量，取值范围为 $[a, b]$ ， $\mu \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n E(X_i)$ 表示这些变量的平均期望。当 $\varepsilon > 0$ 时，有

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| > \varepsilon\right] < 2 \cdot e^{-\frac{2\varepsilon^2}{(b-a)^2}n} \quad (\text{D.6})$$

对于各个变量分布相同的特殊情形，Hoeffding 不等式可由切诺夫界得到利用区间 $[a, b]$ 到区间 $[0, 1]$ 的线性映射。这种特殊情形可用于求定义于较大域上（有界）函数的平均值，特别是当期望的错误概率需要忽略时（在相关参数情况下比任何多项式减少得快）。如果我们能从函数的定义域中采样（并求值），则可得到这样的计算方法。

^① 例如，验证当前证明实际上可应用于 $X_i \in [0, 1]$ 的情形，而非 $X_i \in \{0, 1\}$ ，只需注意 $\text{Var}[X_i] \leq p(1-p)$ 仍成立即可。

D.1.2.4 两两独立和完全独立的采样

在式 (D.6) 中, 为简单起见, 取 $a=0<\mu<b=1$ 。在这种情况下, n 个独立样本给出一个以 ε 偏离期望 μ 的近似概率, 这里表示为 δ , 以 $\varepsilon^2 n$ 指数减少。这样的近似称为 (ε, δ) -近似, 在 $n=O(\varepsilon^{-2} \cdot \log(1/\delta))$ 个样本点的条件下是可以达到的。因此, 样本点数量与 ε^{-1} 多项式相关且与 δ^{-1} 对数相关。相反, 在式 (D.4) 中, n 个两两独立样本的 (ε, δ) 近似要在 $n=O(\varepsilon^{-2} \cdot \delta^{-1})$ 个样本点的条件下才可以达到。这里需要注意, 在两种情况中样本数量都与所需评估精度 ε 多项式相关。相对于两两独立的样本点而言, 完全独立样本点的唯一优势在于其错误概率 δ 对样本数量的依赖性。

D.2 散 列

散列在复杂性理论中广泛使用。典型的应用是将任意 (非结构化的) 几乎均匀的集合映射到结构化的、大小适当的集合。特别地, 散列是以几乎均匀的方式将 $\{0,1\}^n$ 的任意 2^m 子集映射到 $\{0,1\}^m$ 上。

对于基数为 2^m 的固定集合 S , 一一映射 $f_S: S \rightarrow \{0,1\}^m$ 确实存在, 但并不是一个高效的映射 (因为可能需要知道整个集合 S)。显然, 并不存在一个函数 $f: \{0,1\}^n \rightarrow \{0,1\}^m$ 能够以一一对应 (近似一一对应) 的方式把 $\{0,1\}^n$ 的任意 2^m 子集映射到 $\{0,1\}^m$ 上。但是, 对于随机函数 $f: \{0,1\}^n \rightarrow \{0,1\}^m$ 却有这个性质, 对于任意 $S \in \{0,1\}^n$ 的 2^m 子集, f 以相当高的概率把 S 映射到 $\{0,1\}^m$ 上, 且使得值域中每个点在 S 中都不会有太多的原像。问题在于, 一个真正的随机函数不可能具有如此简洁的表达 (更不用说一个高效的求值方法)。因此, 我们要寻找有 “随机映射” 性质 (如以下定义中的条件 (1)) 的函数族, 同时还有简洁的表达和高效的求值算法 (如以下定义中的条件 (2) 和 (3))。

D.2.1 定义

根据上述讨论, 我们考虑这样一个函数族 $\{H_n^m\}_{m \leq n}$, 具有如下性质:

(1) 对于每一个 $S \in \{0,1\}^n$, 在 H_n^m 中均匀选取的函数 h 能够以一种几乎均匀的方式把 S 映射到 $\{0,1\}^m$ 上。例如, 要求对于任意 $|S|=2^m$ 和任意点 y , 关于 h 的选取, 以极高的概率有 $|\{x \in S: h(x)=y\}| \leq \text{poly}(n)$ 。

(2) H_n^m 中的函数有简洁的表达式。例如, 可能要求 $H_n^m = \{0,1\}^{\ell(n,m)}$, ℓ 为某个多项式。

(3) H_n^m 中的函数能够高效求值。也就是存在一个多项式时间算法, 输入函数的表示形式、函数 h (在 H_n^m 中)、字符串 $x \in \{0,1\}^n$, 然后返回 $h(x)$ 。在某些情况下还会对算法制定更严格的条件 (如运行于线性时间)。

条件 (1) 有意留下了疑问。至少要求 $\{x \in S: h(x)=y\}$ 的大小为 $|S|/2^m$ 。应该看到 (2.3 节中), 对条件 (1) 的不同解释可以满足不同的散列函数族。这里我们重点关注 t -元独立的

哈希函数, 定义将在后面给出。

定义 D.1 (t -元独立哈希函数) 如果对于任意 t 个不同域元素 $x_1, x_2, \dots, x_t \in \{0, 1\}^n$ 和任意 $y_1, y_2, \dots, y_t \in \{0, 1\}^m$ 有 $\Pr_{h \in H_n^m} [\wedge_{i=1}^t h(x_i) = y_i] = 2^{-tm}$, 那么, n 比特字符串到 m 比特字符串的 H_n^m 函数族称为 t -元独立的。

也就是说, 均匀选取的 $h \in H_n^m$ 以完全均匀的方式将任意 t 个域元素以完全均匀的方式映射到某个范围。对于 $t \geq 2$, 一个随机的 $h \in H_n^m$ 把两个不同域元素映射为相同象的概率为 2^{-m} 。这样的函数 (函数族) 称为通用的 (Universal), 但我们将主要考虑 t -元独立的更强条件。

D.2.2 构造

以下构造仅仅是对 8.5.1.1 节中的构造提出新的解释 (或者可以把 8.5.1.1 节中的构造看作是以下两个构造的重新诠释)。

构造 D.2 (t -元独立散列) 对于 $t, m, n \in \mathbb{N}$, $m < n$, 考虑以下把 n 比特字符串映射为 m 比特字符串的 H_n^m 函数族。每个 t 长的序列 $\bar{s} = (s_0, s_1, \dots, s_{t-1}) \in \{0, 1\}^{tn}$ 描述了一个函数 $h_{\bar{s}}: \{0, 1\}^n \rightarrow \{0, 1\}^m$, 这样 $h_{\bar{s}}(x)$ 等于二元表达式 $\sum_{j=0}^{t-1} s_j x^j$ 的 m 比特前缀, 其中运算是在含 2^n 个元素的有限域 $GF(2^n)$ 上的运算。

命题 8.24 表明, 构造 D.2 构成了一个 t -元独立哈希函数族。通常使用 $t=2$ 或者 $t=\Theta(n)$ 。为了令构造足够清楚, 我们需要明确说明 $GF(2^n)$, 具体请参考命题 8.24。对于 $t=2$ 时的一种构造可由命题 8.25 中的两两独立生成器得到。回忆一下, Toeplitz 矩阵是一种所有对角线都均匀的矩阵, 亦即若 $T = (t_{i,j})$ 为 Toeplitz 矩阵, 则对所有 i, j 有 $t_{i,j} = t_{i+1,j+1}$ 。

构造 D.3 (另一种两两独立的散列) 对于 $m \leq n$, 设 T 为 $n \times m$ 的 Toeplitz 矩阵, b 为 m 维向量, 考虑这样一个哈希函数族, 其中函数 $h_{T,b}: \{0, 1\}^n \rightarrow \{0, 1\}^m$ 定义为 $h_{T,b}(x) = Tx + b$ 。

命题 8.25 表明, 构造 D.3 构成一个两两独立的哈希函数族。这里 $n \times m$ 的 Toeplitz 矩阵可以指定为 $n+m-1$ 比特, 则其描述的长度为 $n+2m-1$ 比特。另一种可选的构造 (类似于式 (8.18) 且要求为 $m \cdot n + m$ 比特的表达式) 使用的是任意 $n \times m$ 矩阵而非 Toeplitz 矩阵。

D.2.3 剩余 Hash 引理

现在我们研究一下 2.1 节提到的条件下 (如条件 (1)) 的“几乎均匀”。以下给出的引理将明确阐述这一条件 (其中还蕴含了另一种解释——见定理 D.5)。

引理 D.4 令 m, n 为两个整数, 且 $m \leq n$, H_n^m 为两两独立的哈希函数族, $S \subseteq \{0, 1\}^n$, 则对任意 $y \in \{0, 1\}^m$ 和 $\varepsilon > 0$, 除比例不超过 $\frac{2^m}{\varepsilon^2 |S|}$ 的 $h \in H_n^m$ 之外, 对其他 h , 均有

$$(1-\varepsilon) \cdot \frac{|S|}{2^m} < \left| \{x \in S : h(x) = y\} \right| < (1+\varepsilon) \cdot \frac{|S|}{2^m} \quad (\text{D.7})$$

注意: 对于两两独立 (或 1 元独立), $\{x \in S : h(x) = y\}$ 的理想大小为 $\frac{|S|}{2^m}$, 其期望值均匀

分布在 $h \in H_n^m$ 上。这个引理中 h 的比例上界超出了预期 (为 $|h^{-1}(y) \cap S| \neq (1 \pm \varepsilon) \cdot |S|/2^m$)。当然, 这个范围只有在 $|S| \geq 2^m$ (或 $\varepsilon > 1$) 下才有意义。假设 $\varepsilon = \sqrt[3]{2^m/|S|}$, 我们推断对不超过比例 ε 的 $h \in H_n^m$ 有 $|\{x \in S: h(x) = y\}| = (1 \pm \varepsilon) \cdot \frac{|S|}{2^m}$ 。因此, 对几乎所有 $h \in H_n^m$, 每个元素在集合 S 内约有正确数量的 h 原像。

证明: 给定任意集合 $S \subseteq \{0,1\}^n$ 和任意 $y \in \{0,1\}^m$, 我们估算均匀选择的 $h \in H_n^m$ 不满足式 (D.7) 的概率。在先前的概率空间定义随机变量 ζ_x , 如果 $h(x) = y$ 和 $\zeta_x = 0$, 则 $\zeta_x = \zeta_x(h) = 1$, 反之不然。 $\sum_{x \in S} \zeta_x$ 的期望值为 $\mu \stackrel{\text{def}}{=} |S| \cdot 2^{-m}$, 并且我们感兴趣的是这个值偏离期望的概率。利用切比雪夫不等式, 有

$$\Pr \left[\left| \mu - \sum_{x \in S} \zeta_x \right| > \varepsilon \cdot \mu \right] < \frac{\mu}{\varepsilon^2 \mu^2}$$

因为对于两两独立的 ζ_x , 有 $\text{Var} \left[\sum_{x \in S} \zeta_x \right] < |S| \cdot 2^{-m}$, 以及 $E[\zeta_x] = 2^{-m}$ 。引理得证。 ■

一种推广 (称为混合)

引理 D.4 的证明可以简单扩展为证明对于每个 $T \subset \{0,1\}^m$, $\varepsilon > 0$, 不超过比例 $\frac{2^m}{\varepsilon^2 |S| |T|}$ 的 $h \in H_n^m$, 有 $|\{x \in S: h(x) \in T\}| = (1 \pm \varepsilon) \cdot |T| \cdot |S|/2^m$ (提示: 如果 $h(x) \in T$ 和 $\zeta_x = 0$ 时 $\zeta_x = \zeta_x(h) = 1$, 反之则不然)。这个断言在 $|T| \cdot |S| > 2^m / \varepsilon^2$ 和 $m = n$ 条件下有意义, 称为混合属性。

一个极有用的推论

前面提到的引理 D.4 的推广说明了对于任意给定的原像集 $S \subset \{0,1\}^n$ 及任意给定的像集 $T \subset \{0,1\}^m$, H_n^m 中大多数函数关于 S 和 T 都是良好的 (可将 S 的适当部分 (如 $|T|/2^m$) 映射到 T)。在引理 D.4 中蕴含着一个似乎更强的声明, 在合适的集合 S 中大多数 $h \in H_n^m$ 以几乎均匀的方式把 S 映射到 $\{0,1\}^m$ 。这就是以下定理。

定理 D.5 (剩余散列引理) 令 H_n^m 和 $S \subseteq \{0,1\}^n$ 如引理 D.4, 并且定义 $\varepsilon = \sqrt[3]{2^m/|S|}$ 。考虑对于分别在 S 和 H_n^m 均匀分布的随机变量 X 和 H , 则 $(H, H(X))$ 和 (H, U_m) 的统计距离最大为 2ε 。

从而, 对于定理 D.5 中的 X 和 ε 及任意的 $\alpha > 0$, 除了不超过比例 α 的 $h \in H_n^m$ 之外, $h(X)$ 与 U_m 是 $(2\varepsilon/\alpha)$ -接近的^① (利用附录 D.4 中的术语, 可以说 H_n^m 生成了一个强提取器 (这里参数会在后面说明))。

证明: 令 V 表示不满足式 (D.7) 的集合 (h, y) , 并且 $\bar{V} \stackrel{\text{def}}{=} (H_n^m \times \{0,1\}^m) \setminus V$, 则对于每个 $(h, y) \in \bar{V}$, 有

$$\Pr[(H, H(X)) = (h, y)] = \Pr[H = h] \cdot \Pr[h(X) = y] = (1 \pm \varepsilon) \cdot \Pr[(H, U_m) = (h, y)]$$

① 为了证明这一点, 可定义一个随机变量 $\zeta = \zeta(h)$, 满足 ζ 等于 $h(X)$ 与 U_m 的统计距离, 并应用马尔可夫不等式。

另一方面,引理 D.4(对于 $y \in \{0,1\}^m$, 有 $\Pr[(H, y) \in V] \leq \varepsilon$) 和 ε , 我们有 $\Pr[(H, U_m) \in V] \leq \varepsilon$, 则

$$\Pr[(H, H(X)) \in V] = 1 - \Pr[(H, H(X)) \in \bar{V}] \leq 1 - \Pr[(H, U_m) \in \bar{V}] + \varepsilon < 2\varepsilon$$

利用所有这些上界, 通过分离 V 和 $\bar{V}$ 的作用, 得到 $(H, H(x))$ 和 (H, U_m) 的统计差异上界, 定义为 Δ 。特别地, 我们有

$$\begin{aligned} \Delta &= \frac{1}{2} \cdot \sum_{h(x,y) \in H_n^m \times \{0,1\}^m} |\Pr[(H, H(X)) = (h, y)] - \Pr[(H, U_m) = (h, y)]| \\ &\leq \frac{\varepsilon}{2} + \frac{1}{2} \cdot \sum_{(h,y) \in V} |\Pr[(H, H(X)) = (h, y)] - \Pr[(H, U_m) = (h, y)]| \\ &\leq \frac{\varepsilon}{2} + \frac{1}{2} \cdot \sum_{(h,y) \in V} |\Pr[(H, H(X)) = (h, y)] + \Pr[(H, U_m) = (h, y)]| \\ &\leq \frac{\varepsilon}{2} + \frac{1}{2} \cdot (2\varepsilon + \varepsilon) \end{aligned}$$

其中第一个不等式是平凡的 (即对任意非负的 α 和 β , 有 $|\alpha - \beta| \leq \alpha + \beta$), 第二个不等式使用了上述的上限 (即 $\Pr[(H, H(X)) \in V] \leq 2\varepsilon$ 及 $\Pr[(H, U_m) \in V] \leq \varepsilon$)。定理得证。■

定理 D.5 的另一种证明方法

定义 $cp(Z)$ 为随机变量 Z 的碰撞概率, 即 Z 的两个独立样本产生相同结果的概率。另一方面, $cp(Z) \stackrel{\text{def}}{=} \sum_z \Pr[Z = z]^2$ 。定理 D.5 结合以下两个事实说明:

(1) 一个普遍事实。如果 $Z \in [N]$, $cp(Z) \leq (1 + 4\varepsilon^2)/N$, 则 Z 在 $[N]$ 上以 ε 近似服从均匀分布。

我们证明逆否命题: 假设 δ 为 Z 与 $[N]$ 上均匀分布的统计距离, 证明 $cp(Z) \geq (1 + 4\delta^2)/N$ 。定义 $L \stackrel{\text{def}}{=} \{z : \Pr[Z = z] < 1/N\}$, 利用均匀分布下碰撞概率最小的事实为 $cp(Z)$ 规定下界。特别地, 我们分别考虑 L 和 $[N] \setminus L$ 上的均匀分布, 有

$$cp(Z) \geq |L| \cdot \left(\frac{\Pr[Z \in L]}{|L|} \right)^2 + (N - |L|) \cdot \left(\frac{\Pr[Z \in [N] \setminus L]}{N - |L|} \right)^2 \quad (\text{D.8})$$

$$\delta = \rho - \Pr[Z \in L], \quad \rho = |L|/N, \quad \text{式(D.8)的右边等于 } 1 + \frac{\delta^2}{(1-\rho)\rho} \geq 1 + 4\delta^2.$$

(2) $(H, H(X))$ 的碰撞概率最大为 $(1 + (2^m/|S|))/(|H_n^m \cdot 2^m|)$ (只有当 H_n^m 为通用的哈希函数族时这才成立)。

证明方法是直接计算。具体地, 注意: $cp(H, H(X)) = |H_n^m|^{-1} \cdot E_{h \in H_n^m} [cp(h(X))]$, 而 $E_{h \in H_n^m} [cp(h(X))] = |S|^{-2} \sum_{x_1, x_2 \in S} \Pr[H(x_1) = H(x_2)]$, 其和等于 $|S| + (|S|^2 - |S|) \cdot 2^{-m}$, 所以 $cp(H, H(X)) < |H_n^m|^{-1} \cdot (2^{-m} + |S|^{-1})$ 。

这里 $(H, H(X))$ 与 (H, U_m) 是 $\sqrt{2^m/4|S|}$ -接近的, 比定理 D.5 中给出的界限更强。

通过增加独立性来获得更强的均匀性

引理 D.4 中讲到对于哈希函数定义域内的每个点, 对所有哈希函数, 该点等于 S 中原像的数量以高概率近似为期望值。一个更强的条件是以高概率选取哈希函数时, 其定义域内的所有点为 S 中原像的数量近似达到期望值。这可利用 n -元独立哈希函数得到。

引理 D.6 令整数 $m \leq n$, H_n^m 为 n 元独立哈希函数族, $S \subseteq \{0,1\}^n$, 则对于任意 $\varepsilon \in (0,1)$, 除不超过比例 $2^m \cdot (n \cdot 2^m / \varepsilon^2 |S|)^{n/2}$ 的 $h \in H_n^m$ 之外, 式 (D.7) 对所有 $y \in \{0,1\}^m$ 成立。

实际上, 这个引理应该在 $2^m < \varepsilon^2 |S| / 4n$ 条件下使用。特别是, 在 $m = \log_2 |S| - \log_2 (5n / \varepsilon^2)$ 时保证了每个元素都有很大的概率成为 $(1 \pm \varepsilon) \cdot |S| / 2^m$ 个 S 的原像。在参数为 $|S| / 2^m = 5n / \varepsilon^2$, 任何时候 $\varepsilon = 1 / \text{poly}(n)$ 。当然, 这个保证比定理 D.5 中的结论要强。

证明: 以下证明参考引理 D.4 的证明, 利用随机变量 (如 ζ_x) 是 n -元独立的事实。对于 $t = n/2$, 要用到所谓的 $2t^{\text{th}}$ 矩分析, 它是对两两独立抽样的二次矩分析 (附录 D.1.2 中提及) 的推广。与引理 D.4 的证明相同, 我们改变任意 S 和 y , 定义当且仅当 $h(x) = y$ 时

$\zeta_x = \zeta_x(h) = 1$, 令 $\mu = E\left[\sum_{x \in S} \zeta_x\right] = |S| / 2^m$ 以及 $\bar{\zeta}_x = \zeta_x - E(\zeta_x)$, 从马尔可夫不等式开始:

$$\Pr\left[\left|\mu - \sum_{x \in S} \zeta_x\right| > \varepsilon \cdot \mu\right] < \frac{E\left[\left(\sum_{x \in S} \bar{\zeta}_x\right)^{2t}\right]}{\varepsilon^{2t} \mu^{2t}} = \frac{\sum_{x_1, x_2, \dots, x_{2t} \in S} E\left[\prod_{i=1}^{2t} \bar{\zeta}_{x_i}\right]}{\varepsilon^{2t} \cdot (|S| / 2^m)^{2t}} \quad (\text{D.9})$$

利用 $2t$ -元独立性, 发现只有式 (D.9) 中的项没有消失, 这些项里的变量的出现呈多样性。这说明, 只有少于 t 个不同的变量作用于式 (D.9)。接下来, 对于任意 $j \leq t$, 有少于

$\binom{|S|}{j} \cdot (2t!) < (2t! / j!) \cdot |S|^j$ 个项, 这些项有 j 个不同的变量, 而且这些项对于最后总和的作用要少于 $(2^{-m})^j$ 。因此, 式 (D.9) 的上界

$$\frac{2t!}{(\varepsilon |S| / 2^m)^{2t}} \cdot \sum_{j=1}^t \frac{(\varepsilon |S| / 2^m)^j}{j!} < 2 \cdot \frac{2t! / t!}{(\varepsilon^2 |S| / 2^m)^t} < \left(\frac{2t \cdot 2^m}{\varepsilon^2 |S|}\right)^t$$

其中第一个不等式假设 $|S| > n2^m$ (考虑到否则该断言将无意义, 故这个假设是合理的)。这给出了关于固定的 y , 随机选择的 $h \in H_n^m$ 不满足式 (D.7) 的概率上界。对所有 $y \in \{0,1\}^m$ 求联合界, 引理得证。 ■

D.3 采 样

在很多应用环境中, 利用在较大集合上的重复采样来求均值 (或其他统计数据)。^①即给定“赋值”函数 $v: \{0,1\}^n \rightarrow \mathbf{R}$, 希望在不知道定义域中每个点的值 v 的情况下近似求

^① 事实上, 在附录 D.1.2.4 中已经提到这个问题。

$\bar{v} \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{x \in \{0,1\}^n} v(x)$ 。最明显的方法就是在域中随机抽样,在取得样本点 v 的平均值后就可以得

到 $\bar{v}$ 的近似。事实证明,某些“伪随机”序列的样本点可以几乎与真随机样本点序列起相同的作用,因此上述问题实际上与 8.5 节有关。

D.3.1 定义的环境

最重要的是,令 v 的范围有限(否则,不可能有合理的近似)。为了简单起见,我们定义 v 的范围为 $[0,1]$,对于其他问题也可以这样处理。在求近似时主要依靠两个参数,精度 ε 和错误概率 δ 。我们希望得到一个算法,能以至少 $1-\delta$ 的概率得到与正确值相差 ε 范围内的值。这就有了以下的定义。

定义 D.7 (采样器) 采样器是一个随机预言机,输入的参数为长度 n 、精度 ε 和错误概率 δ ,并且预言机可以访问任何一个函数 $v: \{0,1\}^n \rightarrow [0,1]$,以至少 $1-\delta$ 的概率输出一个值,

其与 $\bar{v} \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{x \in \{0,1\}^n} v(x)$ 相差不超过 ε ,即

$$\Pr \left[\left| \text{sampler}^v(n, \varepsilon, \delta) - \bar{v} \right| > \varepsilon \right] < \delta$$

其中概率取自采样器的所有内部随机数。

一个非自适应的采样器包含两个确定性算法:一个是采样生成算法 G ;另一个是估计算法 V 。对于输入 n 、 ε 和 δ ,以及足够长的随机种子,算法 G 产生一个询问序列,记作 $s_1, s_2, \dots, s_m \in \{0,1\}^n$ 。算法 V 在给定相应的 v 值 ($v(s_1), v(s_2), \dots, v(s_m)$) 后输出 $\bar{v}$ 的估计。

我们感兴趣的是,以 n 、 ε 和 δ 为参数的函数的“抽样复杂度”。特别地,考虑 3 个复杂度指标:采样复杂度(即采样器的预言询问次数)、随机性复杂度(即采样器使用的随机种子长度)、计算复杂度(即采样器的运行时间)。当采样器的运行时间为总询问长度的多项式时,认为其是高效的(即同时为采样复杂度和 n 的多项式)。主要讨论高效的采样器,另外,我们感兴趣的是具有最优采样复杂度的采样器。注意:在不考虑采样复杂度时,使随机复杂度最小化毫无意义。

D.3.2 已知结论

注意以下所有的正面成果都是针对非自适应采样器,而对于一般采样器,下界也成立。更多的相关结论详见参考文献[90]中的 3.6.4 节和其中的参考文献。

简单采样器

最直接的方法(即简单采样器)是均匀、独立选择足够多的样本点(询问),并输出这些点的函数平均值。应用切诺夫界有 $O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ 个样本点就够了。以下将指出,简单采样器是具有最优(取决于一个常数因子)采样复杂度的采样器,但使用了太多随机数。

已知任何采样器都需要 $\Omega\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ 个样本,但是采样器做 $s(n, \varepsilon, \delta)$ 次询问需要的随机性复杂度至少为 $n + \log_2(1/\delta) - \log_2 s(n, \varepsilon, \delta) - O(1)$ 。这些采样器的下界是很紧致的(非明确

和效率低下的采样器可以达到下界)。上述事实促使我们寻求改进,重点在于寻求更节省随机数的方式,可以高效地生成采样序列,并与求值算法 V 相结合使用(这里 V 并不一定要取样本点的平均值)。

两两独立采样器

使用两两独立发生器生成样本点,以及采用自然的求值算法(输出样本点平均值),可以极大地节省随机复杂度。特别是,使用随机种子长度为 $2n$, 可以产生 $O(1/\delta\epsilon^2)$ 个两两独立的样本点,(由式(D.4))能够以错误概率 δ 得到足够的精度 ϵ 。因此,两两独立采样器使用 $2n$ 个随机数,而不是简单采样器的 $\Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ 个。另外,对于常数 $\delta > 0$, 两两独立采样器可以同时在样本和随机性复杂度上达到常数因子的最优化。然而,对于较小的 δ (如 $\delta = o(1)$), 这个采样器在采样复杂度上是很浪费的。

平均数中位采样器

一个新的想法需要使用一个相关工具——扩张图上的随机漫游(见 8.5.3 节和附录 E.2)。具体地,将两两独立采样器与扩张图随机漫游发生器(命题 8.29)相结合来构造新的采样器。其中使用顶点集为 $\{0,1\}^{2n}$ 的扩张图上的 t -步漫游来生成 $t = O(\log(1/\delta))$ 个相关种子的序列,其中需调用 t 次两两独立采样器,每次调用利用相应的 $2n$ 比特生成 $\{0,1\}^n$ 上的 $O(1/\epsilon^2)$ 个样本。这种新的采样器称为平均数中位采样器,其输出为 t 次调用两两独立采样器后得到的 t 个值的中位数。在分析这种采样器时,首先要注意上述每次调用返回的值以至少 0.9 的概率与 $\bar{v}$ 为 ϵ -接近。由定理 8.28(还可见习题 8.44),大部分调用返回的值以至少 $1 - \exp(-t) = 1 - \delta$ 的概率是一个 ϵ -接近的近似。因此,这 t 个值的中位数是正确值的 (ϵ, δ) -近似。平均数中位采样器的采样复杂度为 $\Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, 随机性复杂度为 $2n + O(\log(1/\delta))$, 两者都在常数因子范围内达到了最优。

进一步的改进

平均数中位采样器的随机复杂度可以从 $2n + O(\log(1/\delta))$ 提高到 $n + O(\log(1/\delta\epsilon))$, 同时保持最优采样复杂度 $\Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ 。这是通过把两两独立采样器替代为另一种采样器,该采样器在合适的扩张图和样本中选取随机顶点,对其所有邻居采样,并输出平均值。

平均采样器

平均采样器是非自适应采样器,其求值算法为普通算法——即很少输出样本点的平均值。事实上,两两独立的采样器是一个平均采样器,而平均中位数采样器则不是。更有趣的是,平均采样器可以应用于一般非自适应采样器无法应用的场合。平均采样器与随机性提取器有密切联系,对后者的定义及讨论可见附录 D.4。

一个奇怪的观点

回忆一下,非自适应采样器包含一个采样生成算法 G 和一个求值算法 V , 满足对任意的 $v: \{0,1\}^n \rightarrow [0,1]$, 有

$$\Pr_{(s_1, s_2, \dots, s_m) \leftarrow G(U_K)} \left[\left| V(v(s_1), v(s_2), \dots, v(s_m)) - \bar{v} \right| > \varepsilon \right] < \delta \quad (\text{D.10})$$

式中： k 表示采样器的（随机）种子长度。因此，我们或许会把 G 当成一个伪随机数发生器，且易受一类区分器的攻击，这种区分器由固定算法 V 和任意一个函数 $v: \{0,1\}^n \rightarrow [0,1]$ 确定。具体地，假设 m 个样本为均匀、独立分布时， V 工作良好（即 $\Pr \left[\left| V(v(U_n^{(1)}), v(U_n^{(2)}), \dots, v(U_n^{(m)})) - \bar{v} \right| > \varepsilon \right] < \delta$ ），则要求 G 生成的序列满足相应条件（如式 (D.10)）。上述观点的奇怪之处在于，除了平均采样器这个例子，这里考虑的区分器受提取器 V 影响，而习惯上提取器是设计用以帮助 G 欺骗区分器的。^①

D.3.3 碰撞器

碰撞器可以看作是对平均采样器的一种弱化。特别是，在只考虑布尔函数的条件下，碰撞器需要生成一个样本，只要至少有 ε 比例的函数值等于 1，则该样本中包含一个取值为 1 的点。所以，碰撞器就是一个随机算法，输入为参数 n （长度）、精度 ε 和错误概率 δ ，输出为一个 n 比特串列表，使得对于任意密度大于 ε 的集合 $S \subseteq \{0,1\}^n$ ，列表中以至少 $1-\delta$ 的概率包含 S 中至少一个元素。相应的 (ε, δ) -碰撞问题在 8.5.3 节给出了定义。

当然，任何一个非自适应采样器都会产生一个碰撞器（关于参数 n 、 ε 和 δ ）。^②但是，碰撞比评估目标集合密度容易得多， $O(1/\varepsilon)$ 个（两两独立的）随机样本足够以常数概率命中任意密度为 ε 的集合，而为了在精度 ε 内（以常数错误概率）求布尔函数的近似平均值，需要 $\Omega(1/\varepsilon^2)$ 个样本。实际上，附录 D.3.2 中讨论的足够简化的采样器产生的碰撞器 s 其采样复杂度与 $1/\varepsilon$ 成正比（而不是 $1/\varepsilon^2$ ）。

D.4 随机提取器

从比较弱的源（或者有缺陷的源）提取几乎完美的随机数对于随机算法、程序及协议的实际应用至关重要。在分析后者时，假设它们可以访问一个完美的随机源，而实际应用中可能只能访问较弱的随机源。利用随机提取器可以弥补这个缝隙。随机提取器是一个高效的程序，它（在额外随机性的帮助下）可将任意一个弱随机源转化为几乎完美的随机源。因此，随机提取器可以增强随机源的品质。另外，随机提取器还与几个基本问题相关性，以下讨论。

在前面的讨论中回避了一个关键参数，那就是需要从什么等级的弱随机源中提取几乎完美的随机性。关于弱随机源最好作尽可能少的假设。换句话说，就是希望考虑一大类随机源，并且要求随机提取器（简称提取器）对于该类中任何源都“发挥作用”。附录 D.4.1.1 中定义了这样种随机源，但我们首先要提及对于非常受限的随机源来说，不存在能起作用的确定型提取器。^③为了克服这种不可能性，使用以下两种方法：

- ① 采样器与第 8 章中各种随机数发生器的另一个不同点是其目标是使输出序列中“块”的数量（记作 m ）最小化，而非最大化。然而，另一个目标是使块长度（记作 n ）最大化。
- ② 具体地，从参数为 n 、 ε 和 δ 的任意提取器可得到一个参数为 n 、 2ε 和 δ 的碰撞器（为了解释松弛性的要求，只需注意精度 $\varepsilon=1/2$ 的均值是平凡的，而对任意精度（密度） $\varepsilon<1$ ，求碰撞是非平凡的）。对于非适应性采样器，这一断言显然成立，但是对于适应性采样器实际上也成立。注意到，在碰撞器中，适应性并不提供任何优势，因为可以（不失一般性地）假设所有之前的采样都不属于目标集 S 。
- ③ 例如，考虑随机源集合，其输出为 n 比特串，且任意串的出现概率都不大于 $2^{-(n-1)}$ （即在均匀采样下概率值的 2 倍）。

带种子的提取器

第一种方法考虑使用少量随机数的随机提取器。即这些提取器有两个输入：一个短的真随机种子和一个由任意源生成的相对长的序列，该序列属于指定种类的源。这个方法有两种不同的使用方式：

(1) 实际上能访问几乎完美随机源的应用程序，但是从这个源来的随机比特要比从弱（低质量）源来的比特昂贵的多。因此，可以从几乎完美的源那里获取少量高质量比特，并用它们对从弱源（或低质量源）那里得到的较差的比特进行“提纯”。

(2) 在某些应用中（如使用随机算法），可能需要多次调用应用程序，并使用这些应用的“典型”输出（如在确定程序中，可以使用大数表决）。对于这种随机源，可以多次调用应用程序，使得对每个可能的种子 s ，在调用程序时令其输入为 $\text{extract}(s, r)$ ，最后利用这些调用的“典型”输出。事实上，这有点类似于去随机化（见 8.3 节），且这种方法不能用于密码学和/或分布式环境中。

一些独立源

第二种方法考虑确定性提取器，用来从一些（两个）弱随机性独立源中获取样本。这样的提取器在任何环境中都是可行的（包括在加密中），前提是有所需数量的独立弱随机源。

本节重点考虑第一种类型的提取器（即带种子的提取器）。这是由于相关研究的成熟和这个方向与其他复杂性问题的紧密联系。

D.4.1 定义及各种观点

首先给出对应于上述讨论的定义，然后再讨论它与复杂性理论的联系。

D.4.1.1 主要定义

一大类弱随机源有可能不指定输出。即该类的参数^① β 为上界，其中包含所有的源 X ，使得对任意 x ，有 $\Pr[X=x] \leq \beta$ 。此时，认为 X 的最小熵至少是 $\log_2(1/\beta)$ 。事实上，我们提出的随机源是一些随机变量，并假设其分布范围是一些长度为 n 的串，从而 (n, k) 源是一个分布在 $\{0, 1\}^n$ 上的源，且其最少熵为 k 。

(n, k) 源的一个有趣特例是平均分布在 2^k 字符串子集上的源，此时称其为 (n, k) 平面。一个简单而有用的结论是任意 (n, k) 源都是 (n, k) 平面源的凸组合。

定义 D.8 ((n, k) 源提取器)

(1) 算法 Ext 。如果对于每一个 C 中的源 X ， $\text{Ext}(U_d, X)$ 与 U_m 是 ε -接近的，则称 $\{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ 为类 C 在错误概率 ε 下的提取器。如果 C 是 (n, k) 源的类，那么， Ext 称为 (k, ε) 源提取器。

(2) 如果对 C 中的任意源 X ， $(U_d, \text{Ext}(U_d, X))$ 是 ε 接近 (U_d, U_m) 的，则称算法 Ext 是类 C 的错误概率为 ε 的强提取器。 (k, ε) 强提取器的定义与之相似。

利用上述方法将 (n, k) 源“分解”为 (n, k) 平面源，从而 Ext 为一个 (k, ε) 源提取器，当且仅当它是 (n, k) 平面源在错误概率 ε 下的提取器（对于强提取器有同样的结论）。因此，大多

① 回顾一个随机变量 X 的熵定义为 $\sum_x \Pr[X=x] \cdot \log_2(1/\Pr[X=x])$ 。事实上， X 的最小熵等于 $\min_x \{\log_2(1/\Pr[X=x])\}$ ，且上限就是其熵。

数技术分析都针对 (n, k) 平面源进行。例如, 显然对于 $d = \log(n/\varepsilon^2) + O(1)$, 有一个 (k, ε) 源提取器 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^k$ (证明利用了概率方法和所有 (n, k) 平面源集合上的联合界限)。^①

然而, 我们需要的显然是明确的提取器, 即在多项式时间内实现的提取器。注意: 将 n 比特字符串映射为 m 比特字符串的任意两两独立哈希函数族的求值算法构成了错误概率 $\varepsilon = 2^{-(k-m)/2}$ 下的 (强) (k, ε) 提取器 (见定理 D.5)。但这些提取器需要使用较长的种子 (即 $d \geq 2m$, 实际上在构造 D.3 中 $d = n + 2m - 1$)。附录 D.4.2 研究具有对数种子长度 (如 $d = O(\log(n/\varepsilon))$) 的高效 (k, ε) 提取器的构造。但是在此之前, 要给出针对提取器的其他一些观点。

对数种子长度的一个重要注解

由于多种原因, 对数长度种子 (即 $d = O(\log(n/\varepsilon))$) 的情形十分重要。首先, 当利用有缺陷的随机源 (如带种子提取器动机讨论的第二项中) 来仿真一个随机算法时, 计算负载是种子长度的指数。因此, 如果种子长度是对数, 则可在多项式时间内实现对通用概率多项式时间算法的仿真。类似地, 附录 D.4.1.2 和附录 D.4.1.3 中的应用只有当种子长度为对数时才可行。最后要注意对于 (k, ε) 提取器而言, 对数种子长度是一个绝对下限, 不论 $n > k + k^{\Omega(1)}$ ($m \geq 1$ 和 $\varepsilon < 1/2$)。

D.4.1.2 提取器为平均采样器

提取器和平均采样器之间存在紧密联系 (在附录 D.3 的最后提及)。首先要表明, 对于任何平均采样器, 都会产生一个提取器。令 $G: \{0,1\}^n \rightarrow \{0,1\}^m$ 为一个精度为 ε , 错误概率为 δ 的平均采样器的样本生成算法。即在 $\{0,1\}^m$ 中 G 使用 n 比特随机数, 并产生 t 个样本点, 这样对于每个 $f: \{0,1\}^m \rightarrow [0,1]$, 以至少 $1 - \delta$ 的概率, f 在这些点的值的平均值在 $[\bar{f} \pm \varepsilon]$ 的范围内, 这里 $\bar{f} \stackrel{\text{def}}{=} E[f(U_m)]$ 。定义 $\text{Ext}: [t] \times \{0,1\}^n \rightarrow \{0,1\}^m$, 使得 $\text{Ext}(i, r)$ 为 $G(r)$ 产生的第 i 个样本。我们将证明 Ext 是一个 $(k, 2\varepsilon)$ 提取器, 其中 $k = n - \log_2(\varepsilon/\delta)$ 。

用反证法, 假设存在一个 (n, k) 平面源 X , 当 $S \subset \{0,1\}^m$ 时有 $\Pr[\text{Ext}(U_d, X) \in S] > \Pr[U_m \in S] + 2\varepsilon$ ($d = \log_2 t$, $[t] \equiv \{0,1\}^d$)。定义

$$B = \left\{ x \in \{0,1\}^n : \Pr[\text{Ext}(U_d, x) \in S] > (|S|/2^m) + \varepsilon \right\}$$

则 $|B| > \varepsilon \cdot 2^k = \delta \cdot 2^n$ 。定义当 $z \in S$ 时, $f(z) = 1$, 否则, $f(z) = 0$, 我们有 $\bar{f} \stackrel{\text{def}}{=} E[f(U_m)] = |S|/2^m$ 。但是对于每一个 $r \in B$, 样本 $G(r)$ 的 f 平均值比 $\bar{f} + \varepsilon$ 大得多, 这与假设采样器的错误概率为 δ (与精度 δ 有关) 相矛盾。

① 事实上, 一个关键因素是 (n, k) 平面源的数量为 $N \stackrel{\text{def}}{=} \binom{2^n}{2^k}$ 。对于给定的随机平面源, 一个随机函数 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow$

$\{0,1\}^k$ 不是误差为 ε 的提取器的概率上限为 $p = 2^{2^k} \cdot \exp(-\Omega(2^{d+k}\varepsilon^2))$, 因为当选择 2^{d+k} 个随机 k 比特串时, 存在集合 $T \subset \{0,1\}^k$, 这些串中有多于 $((|T|/2^k) + \varepsilon) \cdot 2^{d+k}$ 个与 T 产生碰撞的概率上限为 p 。注意: 对于 $d = \log_2(n/\varepsilon^2) + O(1)$, 有 $N \cdot p \ll 1$ 。事实上, 同样的分析可应用于 m 比特 (而非 k 比特) 的提取。

现在要说明的是，由提取器可以构造平均采样器。令 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$

为一个 (k, ε) 提取器。考虑由 $G(r) = (\text{Ext}(s, r))_{s \in \{0,1\}^d}$ 定义的样本生成算法 G :

$\{0,1\}^n \rightarrow (\{0,1\}^m)^{2^d}$ 。可以证明， G 对应一个平均采样器，该采样器的精度为 ε ，错误概率为 $\delta = 2^{-(n-k-1)}$ 。

仍用反证法，假设存在一个函数 $f: \{0,1\}^m \rightarrow [0,1]$ ，使得对于 $\delta 2^n = 2^{k+1}$ 个字符串 $r \in \{0,1\}^n$ ，样本 $G(r)$ 的 f 函数值的平均值与 $\bar{f} \stackrel{\text{def}}{=} E[f(U_m)]$ 的偏离超过 ε 。不失一般性，假设对于至少一半的 r ，字符串的平均值大于 $\bar{f} + \varepsilon$ ，令 B 表示这些字符串 r 的集合。于是，对于在 B 上平均分布 B 的 X ，且 X 为 (n, k) 源，有

$$E[f(\text{Ext}(U_d, X))] > E[f(U_m)] + \varepsilon$$

这里（在对于所有 z ，有 $|f(z)| \leq 1$ 情况下）与假设 $\text{Ext}(U_d, X)$ 和 U_m 为 ε -接近相矛盾。

D.4.1.3 提取器为随机的高效错误缩减

根据上述讨论，由提取器能构造一种随机性复杂度较低的错误缩减方法。实际上，由布尔函数的例子出发，可以得出错误缩减是采样问题的一个特例（见下表）。特别地，对于一个双边错误的类别程序 A ，考虑函数 $f_x: \{0,1\}^{\rho(|x|)} \rightarrow \{0,1\}$ ，如果 $A(x, r) = 1$ ，则有 $f_x(r) = 1$ ，否则， $f_x(r) = 0$ 。假设 A 为正确的概率至少为 $0.5 + \varepsilon$ ($\varepsilon = 1/6$)，错误归约相当于给予一个精度为 ε ，期望的错误概率 $\delta \ll \varepsilon$ 的布尔函数 f_x 。特别地，任意 (k, ε) 提取器 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^{\rho(|x|)}$ ， $k = n - \log(1/\delta) - 1$ ，假设 2^d 是可行的（如 $2^d = \text{poly}(\rho(|x|))$ ， $\rho(\cdot)$ 代表最初的算法 A 的随机性复杂度）。这里人们感兴趣的是， n 如何变为 $\rho(|x|)$ 和 δ 的函数（代表对应采样器的随机性复杂度）。

使用提取器 $\text{Ext}: \text{poly}(\rho(|x|)) \times \{0,1\}^n \rightarrow \{0,1\}^{\rho(|x|)}$ 的错误缩减

	错误概率	随机复杂性
原始算法	1/3	$\rho(x)$
所到到的算法	δ (可能由 $ x $ 决定)	n ($\rho(x)$ 和 δ 的函数)

当然，对上述问题的回答取决于所使用提取器的质量。特别地，利用定理 D.10 的第一部分，注意到对任意 $\alpha > 1$ ， $\delta > 2^{-\text{poly}(\rho(|x|))}$ 可以得到 $n = O(\rho(|x|) + \alpha \log_2(1/\delta))$ 。另外，对于 $\delta < 2^{-\text{poly}(\rho(|x|))}$ ，错误归约的随机复杂度上界会比由膨胀随机步生成器的 $n = \rho(|x|) + \log_2(1/\delta)$ 上界要好，虽然这里的样本数量比较多（如 $\text{poly}(\rho(|x|)/\delta)$ ，而不是 $O(\log(1/\delta))$ ）。

至于对单边错误概率的约减，可对提取器条件做相应的弱化，称为扩散器 (Disperser)。不严格地，对于 (k, ε) 扩散器，只要求其像（以大于 0 的概率）位于任意密度大于 ε 的集合中即可，而不是要生成一个与均匀分布为 ε -接近的分布。

定义 D.9 (扩散器) 如果对于任意 (n, k) 源 X ， $\text{Dsp}(U_d, X)$ 覆盖至少 $(1 - \varepsilon) \cdot 2^m$ 个点，

则称算法 $\text{Dsp}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ 为一个 (k, ε) 扩散器。另外, 对于每一个规模大于 $\varepsilon 2^m$ 的 $S \subset \{0,1\}^m$, 有 $\Pr[\text{Dsp}(U_d, X) \in S] > 0$ 。

扩散器可以用于减少单边错误, 类似于提取器用于减少双边错误。特别地, 对于前面提到的函数 f_x (假设 $\Pr[f_x(U_{\ell(|x|)}) = 1] > \varepsilon$ 或者 $f_x(U_{\ell(|x|)}) = 0$), 可以利用任意的 (k, ε) 扩散器 $\text{Dsp}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^{\ell(|x|)}$ 来寻找一个点 z 满足 $f_x(z) = 1$ 。实际上, 如果 $\Pr[f_x(U_{\ell(|x|)}) = 1] > \varepsilon$, 则 $\left| \left\{ z: (\forall s \in \{0,1\}^d) f_x(\text{Dsp}(s, z)) = 0 \right\} \right| < 2^k$, 因此使用 n 个随机比特便可令单边错误从 $1 - \varepsilon$ 减少到 $2^{-(n-k)}$ (注意到扩散器与碰撞器 s 是紧密相关的), 与提取器和平均采样器的关系类似。

D.4.1.4 其他观点

用二分图来描述提取器和扩散器十分方便。首先从扩散器开始, 可将任意的 (k, ε) 扩散器 $\text{Dsp}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ 视为一个二分图 $G = (\left(\{0,1\}^n, \{0,1\}^m\right), E)$, 满足 $E = \{(x, \text{Dsp}(s, x))\}$, $x \in \{0,1\}^n$, $s \in \{0,1\}^d$ 。在该图上, 任何 2^k 子集的左边 (如 $\{0,1\}^n$) 顶点有一个相邻的包含至少 $1 - \varepsilon$ 比例的右顶点, 在一些 d 很小 (如 $d = O(\log n / \varepsilon)$) 和 $n \gg k \gg m$, 因为 $m = \Omega(k)$ 的典型例子中这一点尤为突出。而且, 如果 Dsp 可以高效计算, 那么, 这个二分图肯定能构造出来, 给定一个左边顶点, 可以高效地找到所有的邻居节点。任意 (k, ε) 提取器 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ 同样可看作是类似的图, 且具有更强的性质: 左边任意 2^k 个顶点的子集的邻居节点集以几乎均匀的方式包含右边顶点。

一种另类的观点

除了把提取器看作平均采样器以外, 也可以认为其属于伪随机性范畴, 我们还要介绍一个更加另类的观点。具体而言, 将随机性提取器看作是随机算法 (区分器), 设计用于欺骗任意的弱随机源。考虑一个 (k, ε) 提取器 $\text{Ext}: \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$, 其中 $\varepsilon < 1/100$, $m = k = \omega(\log n / \varepsilon)$ 和 $d = O(\log n / \varepsilon)$, 考虑如下类型的区分器 (或测试), 其参数为 $S \subset \{0,1\}^m$ 的子集, 使得与 $S \subset \{0,1\}^m$ 相关联的测试, 记作 T_S , 满足 $\Pr[T_S(x) = 1] = \Pr[\text{Ext}(U_d, x) \in S]$, 也就是说, 对于输入 $x \in \{0,1\}^n$, T_S 均匀选择 $S \in \{0,1\}^d$, 并当且仅当 $\text{Ext}(s, x) \in S$ 时输出 1。以下将证明, 任意 (n, k) 源关于这类区分器都是“伪随机”的, 但充分的“非 (n, k) 源”在这类区分器下不是“伪随机”的。

(1) 对于任意 (n, k) 源 X 及任意 $S \subset \{0,1\}^m$, 测试 T_S 并不能区分 U_n 与 X (即 $\Pr[T_S(X)] = \Pr[T_S(U_n)] \pm 2\varepsilon$), 因为 $\text{Ext}(U_d, X)$ 与 $\text{Ext}(U_d, U_n)$ 为 2ε -接近 (每一个都是 ε -接近 U_m 的))。

(2) 另一方面, 对任意的 $(n, k - d - 4)$ 平面源 Y , 存在一个集合 S , 使得 T_S 能够以至少 0.9 的缝隙从 U_n 中区分 Y (如对于 S , 相当于在 $\text{Ext}(U_d, Y)$ 的支持下, 有 $\Pr[T_S(Y)] = 1$ 和 $\Pr[T_S(U_n)] \leq |S| \cdot 2^{-m} + \varepsilon = 2^{-4} + \varepsilon < 0.1$)。另外, 任意熵低于 $(k/4) - d$ 的源可由该类检测为有

缺陷的（以至少 $2/3$ 的概率）。^①

因此，这类奇异测试将每个 (n, k) 源视为“伪随机”的，而将明显的低熵源（如熵低于 $(k/4) - d$ ）视为非随机的。这种观点极大地扩展了伪随机范式。

D.4.2 构造

回想起我们曾经讨论过提取器的明确构造；即可在多项式时间内计算的函数 $\text{Ext}: \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ 。当然，这里的问题主要还是参数；即令 (k, ε) 提取器的 m 尽可能大，以及令 d 尽可能小。首先要注意，除了“病态”情形之外，^② $m \leq k + d - (2 \log_2(1/\varepsilon) - O(1))$ 和 $d \geq \log_2((n-k)/\varepsilon^2) - O(1)$ 必须满足，无论其明确与否。前面提及的界限实际上是很紧致的；即存在（非明确）的 (k, ε) 提取器，其 $m = k + d - 2 \log_2(1/\varepsilon) - O(1)$ 且 $d = \log_2((n-k)/\varepsilon^2) - O(1)$ 。显然，我们的目标是通过明确的构造来满足这些界限要求。

D.4.2.1 一些已知结论

尽管在这个问题上已有巨大突破（以及有关“最优”明确构造的要求），但尚未到达最终的目标，不过已经很接近了。以下构造就相当接近最优。

定理 D.10(提取器的明确构造) 形如 $\text{Ext}: \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ 的 (k, ε) 提取器在以下情况中存在：

- (1) 对于任意常数 $\alpha > 0$ 和 $\varepsilon > \exp(-k^{1-\alpha})$ ， $d = O(\log n / \varepsilon)$ 和 $m = (1 - \alpha) \cdot (k - O(d))$ 。
- (2) 对于任意常数 $\varepsilon, \alpha > 0$ ， $d = (1 + \alpha) \cdot \log_2 n$ ， $m = k / \text{poly}(\log n)$ 。

第一部分来自参考文献[113]，第二部分则来自参考文献[201]，这些结论都建立在先前工作基础之上（这里没有引用）。另外，我们并没有提及一些无法比较的结论（与参数的不同范围有关）。

我们没有给出定理 D.10 的证明概要，而是给出了一个更弱结论的证明概要，其中对于 $d = O(\log n)$ 和 $m = n^{\Omega(\gamma)}$ 情形给出了一种明确的 $(n^\gamma, \text{poly}(1/n))$ 提取器，其中 $\gamma > 0$ 是一个任意小的常数。事实上，附录 D.4.2.2 对于该结论所依据的概念（以及该领域的后续研究）进行了综述。

D.4.2.2 伪随机性之间的联系

本节将总结有关于提取器和某些伪随机生成器的联系。这个联系由 Trevisan^[222]发现，令人惊奇的是，它现在正走向一个非标准方向：把某些伪随机生成器转换为提取器。本书一直秉持一个观点（特别是在 7.1.2 节的结尾），即可计算的对象比对应的信息论对象要复杂得多。因此，如果提取器和伪随机生成器是有关联的（在参考文献[222]以前没有疑问），那么，在看到信息论对象的时候，不要期望这种关联有助于提取器的构造。当然，这种关联的发现的确对于提取器研究产生了突破。^③

① 对任意这类源 Y ，分布 $Z = \text{Ext}(U_d, Y)$ 的熵至多为 $k/4 = m/4$ ，从而与 U_m 为 0.7-接近（且与 $\text{Ext}(U_d, U_n)$ 为 2/3-接近）。 Z 与 U_m 的统计距离下限可用反证法证明：如果 Z 与 U_m 为 δ -接近，则其熵至少为 $(1 - \delta) \cdot m - 1$ （如利用费诺不等式，见参考文献[63]中的定理 2.11.1）。

② 即 $\varepsilon < 1/2$ 及 $m > d$ 的情形。

③ 注意到，一旦很好地理解了这种联系，便走上了“正确”方向：从提取器转向伪随机数发生器。

教学注释: 这里假设读者熟悉伪随机数发生器, 特别是 Nisan-Wigderson 发生器(见 8.3.2.1 节)。

但是在描述这种关联之前, 需要先说明所使用的符号。注意到, 伪随机生成器有一个单独的输入(如种子), 而提取器有两个输入(n 比特长的源和 d 比特长的种子)。但是再看 Nisan-Wigderson 生成器, 注意到这个构造可以看作有两个输入: 一个 d 比特长的种子和以 d' 比特串为源的“困难”断言(其中 $d' = \Omega(d)$)。现在, 一个好的方法是利用 n 比特长的源作为对(最坏情况)困难断言(真值表)的描述(实际上意味着令 $n = 2^{d'}$)。^①关键在于即使源只有较弱的随机性, 我们还是期望在最坏的情况下它能够表示一个断言。

回顾前面提及的构造, 只要是从困难谓词开始, 总能产生一个伪随机数发生器。在本附录中, 如果不考虑计算上的限制, 则可以假设伪随机性能够抵抗任何区分器, 并且这里伪随机意味着与均匀分布统计接近(字符串足够长, 以 ℓ 表示)。直观地说, 这只有在观察序列(即源和种子随机性数量)在比较短的情况下成立。因此, 在这种方式下还是可以得到一个好的提取器的。

现在把希望转换为现实, 这需要证明(以下将给出)。再看 Nisan-Wigderson 生成器, 注意到, 在证明这类生成器的不可区分性时, 当给出了到任意区分器的预言访问, 证明中提供了计算当前谓词的黑盒程序。特别地, 定理 7.19 和定理 8.18 的证明中(对于 $\ell = 2^{\Omega(d')}$ 时成立), ^②黑盒过程是由一个相对较小的电路实现的。因此, 在提取器或生成器 ℓ 比特长输出的基础上, 这个过程包含了相当小的信息。

特别地, 对于一些固定多项式 p , 在这个过程中加密的信息量上限达到 $b = p(\ell)$, ^{def}虽然该过程假设能准确计算每个输入上当前的断言。即数据量是假设完全由当前的断言决定的, 对于 n 比特长的源来说也是一样的, 因此, 如果源的最小熵超过了 b , 则利用 b 比特信息是无法完全确定的。前面提到的构造构成了一个 $(b + O(1), 1/6)$ 提取器(输出为 $\ell = b^{\Omega(1)}$ 比特), 这里常数 $1/6$ 仅仅在定理 8.18 证明中使用(对于如果 $b = n^{\Omega(1)}$ 时如何还存在争论)。注意到, 该提取器使用的种子长度为 $d = O(d') = O(\log n)$ 。这种方法可以扩展以构造 $(k, \text{poly}(1/k))$ 提取器, 其输出 $k^{\Omega(1)}$ 比特使用的种子长度 $d = O(\log n)$, 如果 $k = n^{\Omega(1)}$ 。

注意到, 以上只描述了 Nisan-Wigderson 生成器的两个抽象属性: 事实上, 这个生成器使用了任意最坏情况下的困难断言作为黑盒子; 分析时, 则使用任意区分器作为黑盒子。特别是, 我们把不可近似的最坏情况的困难问题作为伪随机生成器的一部分。另一种方法则是自包含的, 将定理 7.19 中的放大步骤用这里的直接方法取代, 并在构造 8.17 中直接插入得到的谓词。这样做的优点是使用了一个更简单的放大(因为在信息论条件下的放大比在计算条件下简单), 并且获得透明的构造和分析(分别对应于构造 8.17 和定理 8.18)。

另一种方法

上述分析将一个通用区分器转换为一个程序, 能正确计算每个输入的当前断言, 这就完全确定了该断言。因此, 如果某个随机源对于提取器而言是“不好”的, 则由该程序可以获取的信息量上界可得到源的可能输出数量上界。在另一种方法中, 把通用的区分器转换为只

① 事实上, 为了适应当前内容, 我们对某些符号做了修改。在构造 8.17 中, 种子长度记作 d , 断言的输入长度记作 m 。

② 回顾, $n = 2^{d'}$, $\ell = 2^{\Omega(d')}$ 的限制蕴含了 $\ell = n^{\Omega(1)}$ 。

能计算当前谓词的程序；也就是说，程序产生一个函数，与谓词相当接近。如果可能的谓词是相互远离的，则可得到期望的界限（对应于这些谓词的“不好”源的输出数量）。因此，主要思想是将 n 比特源编码成码长为 $n' = \text{poly}(n)$ 、相关距离为 $0.5 - (1/n)^2$ 的纠错码，并利用得到的码字作为构造 8.17 中谓词的真值表。^① 这样一个码（有相应的高效译码算法）的确存在（见附录 E.1.2.5），且使用它们的好处是每 n' 比特的串（由上述近似程序能获得的信息确定）可以与最多 $O(n^2)$ 个码字^②（对应于可能的谓词）为 $0.5 - (1/n)$ 接近的。因此，每个求近似的程序可以排除至多 $O(n^2)$ 个可能的谓词（即源的输出）。总结一下，得到的提取器将 n 比特输入 x 转换为码字 $x' \in \{0,1\}^{n'}$ ，视为一个 $\{0,1\}^{d'}$ 上的断言（其中 $d' = \log_2 n'$ ），且在 d 比特长种子的 ℓ 个映射下计算该断言，这些映射是由相应的集合系统确定。这个分析缩小了定理 8.18 的证明，且得到“不好”的随机源输出数量的上界 $2^{O(\ell^2)} \cdot O(n^2)$ ，其中 $O(\ell^2)$ 是源的输出中，对应于与任意给定近似程序为 $0.5 - (1/n)$ 接近的码字的输出数量上界。

D.4.2.3 推荐阅读

有兴趣的读者可以参考 Shaltiel^[200] 的研究。其中包含这个领域内的全面介绍，也包括各种构造思想的概述。特别是，这个研究还描述了在发现伪随机性联系以前使用的一些方法，联系本身（从当中产生的一些构造）以及“第三代”构造。

以上综述早于后来提出的一些（提取器的）构造，这些构造利用对数长度的种子提取最小熵的常数部分（见定理 D.10 的第一部分）。这类构造最早见于参考文献[159]，参考文献[113] 对其进行了改进。我们建议读者阅读参考文献[113]，其中对已知的最好提取器（在几乎所有参数设置下）有一个自包含的描述。

① 事实上，使用这种纠错码来代替定理 7.19 中的困难放大步骤。

② 见附录 E.1.4。

附录 E

明确的构造

富人进天堂比骆驼穿过针眼还要困难。
马修斯，19：24

复杂性理论给出了明确构造的直观定义。也针对构造目标的难易程度，对构造的明确性进行分层。

最基本的明确性需考虑完全构造的复杂度（如打印有限函数的真值表所花费的时间）。此时，明确性往往意味着在对象完整描述长度的多项式时间内输出该描述。更高水平的明确性还需考虑回答关于该对象询问的复杂度（如对于给定输入计算一个函数所花费的时间）。在这种情况下，（强）明确性通常意味着在多项式时间内回答询问。

本附录回顾纠错码和扩张图的明确构造，以阐述上述问题。正文的不同部分都使用了这些构造。

内容提要

本附录简要回顾与复杂性密切相关的几种纠错码和扩张图的构造。从纠错码开始，回顾几种常用的构造，最突出的是“良好”二进制码（即达到常数相关距离和常数码率）的构造。后者可以通过将 Reed-Solomon 码与一个“适度明确”的“良好”二进制码（信息位较少）进行级联而构造。我们还简要回顾本地可测试码及本地可译码的概念，并提出一种有用的“查表译码界”（即接近任意序列的码字数量上界）。

对于扩张图，回顾了扩张的两个标准定义（分别从组合论及代数观点），以及扩张图与其上的（单步和多步）随机漫游相关的两种属性。我们还解释了图的明确性的两种水平，对应于上述的基本明确性和较强的明确性。最后，回顾了扩张图的两种明确构造。

E.1 纠错码

本节主要讨论与本书关系最密切的编码理论问题。有兴趣的读者可以见参考文献[217]以获得对编码理论更深入的理解。标准教科书如参考文献[163]，也涵盖了编码理论结构方面的内容，这是该领域传统上的重点内容。

E.1.1 基本概念

粗略地说，纠错码是字符串到更长字符串上的映射，使得任何两个不同的字符串被映射

到相应的一对距离很远的（不仅仅是不同的）字符串。具体来说，如果对每个 $x \neq y \in \{0,1\}^k$ 使得 $C(x)$ 和 $C(y)$ 至少有 d 位不同，则 $C: \{0,1\}^k \rightarrow \{0,1\}^n$ 是距离为 d 的（二进制）码。事实上， k 、 n 与 d 的关系是要考虑的主要因素：编码的典型目标是让码间距离较大（即 d 要足够大），同时不引入太多冗余^①（即令 n 尽可能小）。

将上述定义扩展到任意字母表 Σ 是有价值的。具体而言，对于 $x \in \Sigma^m$ ，用 x_i 表示 x 的第 i 个符号（即 $x = x_1 x_2 \cdots x_m$ ），并在考虑 Σ 上的码（即 Σ -序列映射到 Σ 序列）。如果对于每个 $x \neq y \in \Sigma^k$ 存在 $\{i: C(x)_i \neq C(y)_i\} \geq d$ ，则映射（码） $C: \Sigma^k \rightarrow \Sigma^n$ 的距离为 d 。 $\{C(x): x \in \Sigma^k\}$ 的成员称为码字（在某些教材中这个集合本身被称为一个码）。

Σ 上 n 长序列集合的一个度量称为汉明距离。当 $y, z \in \Sigma^n$ 时， y 与 z 之间的汉明距离定义为取值不同的位置数（即 $\{i: y_i \neq z_i\}$ ）。序列的汉明重量定义为非零元素的数目（假定 Σ 中有一个元素被视为零）。典型地， Σ 与加法群相关联，且在此情况下 y 和 z 之间的距离等于 $w = y - z$ 的汉明重量，其中 $w_i = y_i - z_i$ （对所有 i ）。

渐近性

我们实际上考虑的是码字的无限族，即 $\{C_k: \Sigma_k^k \rightarrow \Sigma_k^{n(k)}\}_{k \in S}$ ，其中 $S \subseteq \mathbb{N}$ （通常 $S = \mathbb{N}$ ）（注意：允许 Σ_k 取决于 k ）。如果对任意 $k \in S$ ， C_k 的距离为 $d(k)$ ，称这一个族的距离为 $d: \mathbb{N} \rightarrow \mathbb{N}$ 。显然， $n = n(k)$ （称为块长度）和 $d(k)$ 均与 k 相关，而目标是让其线性相关（即 $n(k) = O(k)$ 且 $d(k) = \Omega(n(k))$ ）。在此情况下，涉及到码的相对码率（即常量 $(k/n(k))$ 和相对距离（即常量 $d(k)/n(k)$ ）。通常会提及序列间的相对距离。例如，对于 $y, z \in \Sigma^n$ ，如果有 $|\{i: y_i \neq z_i\}| \leq \varepsilon n$ （或者 $|\{i: y_i \neq z_i\}| \leq \varepsilon n$ ），则称 y 和 z 是 ε -接近的（或者 ε -远离的）。

明确性

温和的明确性是指在其长度（是 k 的指数）的多项式时间内构造出所有码字。更标准的明确性，指的是产生一个特定的码字（即给定 x 生成 $C(x)$ ），这与下面的编码任务一致。更强的明确性概念涉及有关编码的其他计算问题（见后面内容）。

计算问题

与纠错码相关的最基本的计算任务是编码和译码（在噪声环境下）。编码的定义非常直接（即从 $x \in \Sigma_k^k$ 到 $C_k(x)$ 的映射），并且必须有一个高效的算法在 $\text{poly}(k, \log|\Sigma_k|)$ 时间内计算 $C_k(x)$ 中每个符号。^② 定义译码任务时，注意到“最小距离译码”（即给定 $w \in \Sigma_k^{n(k)}$ ，找到使 $C_k(x)$ 最接近（在汉明距离意义上） w 的 x ）仅仅是一种可能性。关于距离为 d 的码，有两个相关的计算问题。

唯一译码：给定 $w \in \Sigma_k^{n(k)}$ ，其与某个码字 $C_k(x)$ 的汉明距离小于 $d(k)/2$ ，求 $C_k(x)$ 相应的译码（即找到 x ）。

当然，该任务是一个良好定义，因为不可能有两个不同的码字与 w 的汉明距离均小于 $d(k)/2$ 。

列表译码：对于给定的 $w \in \Sigma_k^{n(k)}$ 和参数 $d' \geq d(k)/2$ ，输出与 w 的汉明距离不超过 d'

① 注意：为了让距离为 d ，一种平凡方法是把每个符号重复 d 次。这种（“重复”）码满足 $n = d \cdot k$ ，而我们的目标是令 $n \ll d \cdot k$ 。事实上，我们将看到，可以同时满足 $n = O(k)$ 和 $d = \Omega(k)$ 。

② 这种定义在编码理论中不是最普遍的，但在本书中很自然。一方面，这种定义对于超多项式的分组长度也是可用的。另一方面，这种定义不支持这样的编码算法，该算法比单独计算码字中每个比特要快。

的所有 $w \in \sum_k^k$ 的列表。

一般考虑 $d^* < d(k)$ 的情况。参见附录 E.1.3 对与一个普通序列具有特定距离的码字数量上界的讨论。

附录 E.1.2 中还讨论了另外两个计算任务。

线性码

将 Σ_k 视为有限域, 如果一咱编码方案 $C_k: \Sigma_k^k \rightarrow \Sigma_k^{n(k)}$ 满足 $C_k(\mathbf{x}+\mathbf{y})=C_k(\mathbf{x})+C_k(\mathbf{y})$, 则称其为线性码, 其中 $\mathbf{x}$ 和 $\mathbf{y}$ (或 $C_k(\mathbf{x})$ 和 $C_k(\mathbf{y})$) 是 Σ_k 上的 k 维 (或 $n(k)$ 维) 向量, 运算也在相应向量空间上进行。线性码的一个有用特性是其距离等于除 $C_k(0^k)$ 以外码字的最小汉明重量。也就是说, $\min_{\mathbf{x} \neq \mathbf{y}} \{|\{i: C_k(\mathbf{x})_i \neq C_k(\mathbf{y})_i\}|\}$ 等于 $\min_{\mathbf{x} \neq 0^k} \{|\{i: C_k(\mathbf{x})_i \neq 0\}|\}$ 。另一个有用的特性是编码完全可由一个称为生成矩阵的 $k \times n(k)$ 矩阵来规定。生成矩阵由 Σ_k^k 的一些固定基组成。也就是说, 全体码字集合可以由生成矩阵行向量的所有不同线性组合来生成。

E1.2 几种流行的码

重点讨论可明确构造的码, 即存在高效的编码和译码算法的形如 $\{C_k: \Sigma_k^k \rightarrow \Sigma_k^{n(k)}\}_{k \in S}$ 的码(族)。在给出几个这种码之前, 先考虑一种非明确的构造。

命题 E.1 (随机线性码的距离) 令 $n, d, t: \mathbf{N} \rightarrow \mathbf{N}$, 满足对任意足够大的 k , 有

$$n(k) \geq \max \left(2d(k), \frac{k+t(k)}{1-H_2(d(k)/n(k))} \right) \quad (\text{E.1})$$

其中 $H_2(\alpha) \stackrel{\text{def}}{=} \alpha \log_2(1/\alpha) + (1-\alpha) \log_2(1/(1-\alpha))$, 则对所有足够大的 k , 从 $\{0, 1\}^k$ 到 $\{0, 1\}^{n(k)}$ 的随机线性变换以高概率构成距离为 $d(k)$ 的码。

于是, 对任意常数 $\delta \in (0, 0.5)$, 存在常数 $\rho > 0$ 和一个相关距离为 δ 的码 $\{C_k: \{0, 1\}^k \rightarrow \{0, 1\}^{k/\rho}\}_{k \in \mathbf{N}}$ 的无限族。具体地, $\rho = (1-H_2(\alpha))/c$ 。

证明: 考虑一个随机选择的 $\text{GF}(2)$ 上的 $k \times n(k)$ 阶生成矩阵, 并考虑该矩阵生成一个距离不超过 $d(k)$ 的线性码的概率上界。对生成矩阵行的所有 2^k-1 种线性组合应用联合界, 对于每种组合, 计算产生汉明重量小于 $d(k)$ 的向量的概率。观察到这些线性组合在 $\{0, 1\}^{n(k)}$ 上均匀分布, 因而汉明重量不超过 $d(k)$ 的概率为

$$p \stackrel{\text{def}}{=} \sum_{i=0}^{d(k)-1} \binom{n(k)}{i} \cdot 2^{-n(k)}$$

显然, 对于 $d(k) \leq n(k)/2$, 有 $p < d(k) \cdot 2^{-(1-H_2(d(k)/n(k))) \cdot n(k)}$, 但实际上 $p \leq 2^{-(1-H_2(d(k)/n(k))) \cdot n(k)}$ 也成立 (如利用参考文献[11]中的引理 14.6.3)。利用 $(1-H_2(d(k)/n(k))) \cdot n(k) \geq k+t(k)$, 命题得证。 ■

E1.2.1 命题 E.1 的温和明确版本

注意: 命题 E.1 产生一个寻找距离为 d 的线性码的 (确定性) $(k \cdot n(k))$ 次指数算法。时间界可以提高到 $(k+n(k))$ 的指数级, 可以选择逐一选择生成矩阵的行, 确保当前行的所有非空线性组合的重量不小于 $d(k)$ 。注意: 命题 E.1 的证明可适于这样的断言: 只要少于 k 行, 就会以较高的概率随机选择下一行。当 $n(k)=O(k)$ 时, 可以得到一种算法, 运行时间是码的规模 (即码字数目) 的多项式。当然, 这种温和的明确性对大多数的编码应用程序来说是不够的, 但是在 E.1.1.5 中将会很有用。

E.1.2.2 Hadamard 码

Hadamard 码是 $\{0,1\} \cong \text{GF}(2)$ 上最长的 (非重复) 线性码。也就是说, $x \in \{0,1\}^k$ 映射为各个位的所有 $n(k)=2^k$ 种线性组合的全部序列 (即码字中的位置与 k 比特的串相关联, x 的码字中的位置 $\alpha \in \{0,1\}^k$ 取值 $\sum_{i=1}^k \alpha_i x_i$)。可以验证, 每个非零码字的重量为 2^{k-1} , 且这种编码的相关距离为 $d(k)/n(k)=1/2$ (尽管分组长度 $n(k)$ 是 k 的指数级)。

在计算方面, 注意到编码是很容易的。至于译码, 在定理 7.7 的证明一开始的讨论中提供了一个唯一译码的快速概率算法, 而定理 7.8 给出了一个列表译码的快速概率算法。

Hadamard 码在 PCP 定理的证明起到了关键作用。

一种建议的长码

最长 (非重复) 二进制码 (参考文献[29]中提出的长码) 被广泛用于 “高级的” PCP 系统 (如参考文献[116, 117]) 的设计。在这种码中, k 比特长的字符串 x 被映射到的 $n(k)=2^{2^k}$ 值的序列, 分别对应于不同的布尔函数在输入为 x 时的值; 即码字的位置与布尔函数相关, 使得 x 的码字中与 $f: \{0,1\}^k \rightarrow \{0,1\}$ 相关的位置能够取 $f(x)$ 的值。

E.1.2.3 Reed-Solomon 码

Reed-Solomon 码被定义为一个非二进制字母表, 它与 n 个元素的有限域 $\text{GF}(n)$ 相关。对于任意 $k < n$, 考虑 $\text{GF}(n)$ 上的单变量 $k-1$ 次多项式到所有域元素上多项式值的映射。也就是说, $p \in \text{GF}(n)^k$ (可看成多项式) 被映射为序列 $(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_n))$, 其中 $\alpha_1, \alpha_2, \dots, \alpha_n$ 是 $\text{GF}(n)$ 中元素的枚举。^① 这种映射称为参数为 k 和 n 的 Reed-Solomon 码, 且其距离为 $n-k+1$ (因为任意 $k-1$ 次非零多项式在至少 k 个点取值为 0)。事实上, 该码是一种线性码, 因为 $p(\alpha)$ 是 $p_0, p_1, \dots, p_{k-1}$ 的一种线性组合, 其中 $p(\zeta) = \sum_{i=0}^{k-1} p_i \zeta^i$ 。

Reed-Solomon 码提供了具有常数码率和常数相对距离的码的无限族 (如取 $n(k)=3k$ 和 $d(k)=2k$), 但是字母表的大小随着 k 改变 (或者 $n(k) > k$)。已知有唯一译码和列表译码的高效算法 (见参考文献[216])。这些计算任务对应于多项式在所有点的外部插值。

E.1.2.4 Reed-Muller 码

Reed-Muller 码将一元多项式的 Reed-Solomon 码推广到多变量多项式。同样地, 码表可以是任何有限域, 特别是可以为二元域 $\text{GF}(2)$ 。Reed-Muller 码 (和其变形) 广泛应用于复杂性理论。例如, 它是构造 7.11 和 9.3.2.2 节结尾部分所构造的 PCP 的基础。这些码的相关属性在于, 在一个合适的参数集合下满足 $n(k) = \text{poly}(k)$, 并允许超级快速 “码字测试” 和 “自纠错” (附录 E.1.2 将讨论)。

对于任何素数幂 q 及参数 m 与 r , 考虑这样一个集合 $P_{m,r}$, 表示 $\text{GF}(q)$ 上次数不超过 r 的 m 变量多项式。 $P_{m,r}$ 中每个变量被表示为所有相关的单项式的系数 $k = \log_q |P_{m,r}|$, 其中当 $r < q$ 时有 $k = \binom{m+r}{m}$ 。对于码 $C: \text{GF}(q)^k \rightarrow \text{GF}(q)^n$, 其中 $n = q^m$, 把总次数不超过 r 的 m 变量多项式映射到 q^m 个点的取值。即总次数不超过 r 的 m 变量多项式映射为序列值 $(p(\bar{\alpha}_1), p(\bar{\alpha}_2), \dots, p(\bar{\alpha}_n))$, 其中 $\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_n$ 是 $\text{GF}(q)$ 上的所有 m 元组的正则枚举。这个码字的相对距离是下界 $(q-r)/q$ 。

^① 作为替代, 还可将 $(v_1, v_2, \dots, v_k) \in \text{GF}(n^k)$ 映射到 $(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_n))$, 其中 p 为唯一的 $k-1$ 次单变量多项式, 满足 $p(\alpha_i) = v_i$, $i = 1, 2, \dots, k$ 。注意: 这种修改可归结为生成矩阵的一种线性变换。

在典型应用中, 令 $r = \Theta(m^2 \log m)$, $q = \text{poly}(r)$, 能使 $k > m^m$ 和 $n = \text{poly}(r)^m = \text{poly}(m^m)$ 成立。于是, 有 $n(k) = \text{poly}(k)$, 但 $n(k) = O(k)$ 不成立。附录 E.1.3 中我们将看到其优势(与 Reed-Solomon 码相比)在于, 码字测试和自纠错的复杂度与 $q = \text{poly}(\log n)$ 相关。实际上, 在大多数复杂的应用程序中, 只使用次数为 $r' = r/m$ 的 m 变量多项式的一种变形。在这种情况下, 更倾向于一种类似于脚注的表达: 信息被视为一个函数 $f: H^m \rightarrow \text{GF}(q)$, 其中 $H \subset \text{GF}(q)$ 的规模为 $r'+1$, 通过对扩张函数 f 的 r' 次 m 变量多项式在 $\text{GF}(q)^m$ 上所有点求值来编码(见构造 7.1)。

E.1.2.5 恒定相对距离和恒定码率的二进制码

我们希望构造具有恒定相对距离和恒定码率的二进制码。命题 E.1 断言这样的代码存在, 但是并没有提供一个明确的构造。Hadamard 码是明确的, 但没有恒定的码率(至少可以这样说(因为 $n(k) = 2^k$))。^① Reed-Solomon 码拥有恒定的相对距离和恒定码率, 但使用了非二进制字母表(随着 k 线性增长)。通过使用级联方法^[78]可以达到预期的构造(实际上, 级联码可被视为在 9.3.2.2 节提出的证明组合方法的简单版本)。

直观地说, 为构造级联码, 首先可对信息编码, 信息被视为较大字母表上的序列, 将其变成某个码字, 再对得到的结果编码, 后者可视为一个较小字母表上的序列。正式地说, 考虑 $\Sigma_1 = \Sigma_2^{k_2}$ 和两个码 $C_1: \Sigma_k^{k_1} \rightarrow \Sigma_1^{n_1}$ 与 $C_2: \Sigma_k^{k_2} \rightarrow \Sigma_1^{n_2}$ 。 C_1 和 C_2 的级联码, 将 $(x_1, x_2, \dots, x_2 \in \Sigma_1^{k_1} \equiv \Sigma_2^{k_1 k_2})$ 映射到 $(C_2(y_1), C_2(y_2), \dots, C_2(y_{n_1}))$, 其中 $(y_1, y_2, \dots, y_{n_1}) = C_1(x_1, x_2, \dots, x_{k_1})$ 。

注意到, 如果 C_1 和 C_2 拥有恒定的码率和相对距离, 则生成的码 $C: \Sigma_2^{k_1 k_2} \rightarrow \Sigma_1^{n_1 n_2}$ 也有这些性质。级联码的编码过程非常简单。为了对损坏的码字译码, 把输入看作是长为 n_2 的块序列, 其中每块是 Σ_2 上长为 n_2 的序列。将 C_2 的译码器应用于每个块, 可以获得 Σ_1 上的 n_1 个序列(每个长为 k_2), 并将每个这样的序列作为 Σ_1 上的符号。最后, 将 C_1 的译码器应用于所生成的长为 n_1 的序列 (Σ_1 上), 将长为 k_1 的序列作为 Σ_2 上的长为 $k_1 k_2$ 的序列来解释。关键在于, 如果 $w \in \Sigma_1^{n_1 n_2}$ 是 $\varepsilon_1 \varepsilon_2$ -接近于 $C(x_1, x_2, \dots, x_{k_1}) = (C_2(y_1), C_2(y_2), \dots, C_2(y_{n_1}))$ 的, 那么, w 的块中至少有 $(1-\varepsilon_1) \cdot n_1$ 个与相应的 $C_2(y_i)$ 是 ε_2 -接近的。^②

继续考虑级联码, 以 Reed-Solomon 码 $C_1: \text{CF}(n_1)^{k_1} \rightarrow \text{CF}(n_1)^{n_1}$ 作为大码, 取 $k_2 = \log_2 n$, 以命题 E.1 的温和明确性版本 $C_2: \text{CF}(n_1)^{k_2} \rightarrow \text{CF}(n_1)^{n_2}$ 作为小码。令 $n_1 = 3k_1$, $n_2 = O(k)$, 于是, 级联码为 $C: \{0, 1\}^k \rightarrow \{0, 1\}^n$, 其中 $k = k_1 k_2$, $n = n_1 n_2 = O(k)$ 。而且, C_2 的编码和译码都能在 $\exp(k_2) = \text{poly}(k)$ 时间内实现。于是, 可得如下定理:

定理 E.2 (一个明确的好码) 对于常量 $\delta, \rho > 0$, 存在码率为 ρ 、相关距离至少为 δ 的明确的二进制码族。即存在一个多项式时间的(编码)算法 C 使得 $|C(x)| = |x|/\rho$ (对每个 x) 和一个多项式时间的(译码)算法 D 使得对每个 $\delta/2$ 接近于 $C(x)$ 的 y 有 $D(y) = x$ 。而且, C 是线性码。

C 的线性度可通过扩域 $F = \text{CF}(2^{k_2})$ 上 Reed-Solomon 码来调整, 并注意到, 这个码产生 $\text{GF}(2)$ 上的线性变换。特别地, 当视为 $\text{GF}(2)$ 上的 k_2 维空间向量时, F 上满足 $a \in F$ 的多项式的值 p 可以由一个线性变换的 p 的系数来获得。

临近二分之一的相对距离

从 Reed-Solomon 码相对距离 δ_1 和一个短码 C_2 的相对距离 δ_2 , 可以得到一个相对距离为

① 二元 Reed-Muller 码也无法同时具有常数的相对距离和常数码率。

② 这一观察提供了从部分错误中的唯一译码, 其中错误是与两个原始码相关的(错误)片段之积。关于级联码的唯一译码性更强的结论可基于更精细的分析得到(见参考文献[78])。

$\delta_1\delta_2$ 的级联码。要注意的是, 对于常量 $\delta_1 < 1$, 存在一个相对距离为 δ_1 且码率恒定 (即 $1-\delta_1$) 的 Reed-Solomon 码 $C_1: CF(n_1)^{k_1} \rightarrow CF(n_1)^{n_1}$ 。如果不考虑恒定码率, 可以从分组长度为 $n_1(k_1) = \text{poly}k_1$ 、距离为 $(n_1(k) - k) \cdot n_1(k)/2$ 的 Reed-Solomon 码开始, 再以一个 Hadamard 码 (用 $\{0, 1\}^{[n_1(k)]}$ 来编码 $[n_1(k)]$) 作为小码 C_2 。这样会产生一个分组长度为 $n(k) = n_1(k)^2$ 、距离为 $(n_1(k) - k) \cdot n_1(k)/2$ 的 (级联的) 二进制码。而且, 所得到的明确性码的相对距离约为 $\left(\frac{1}{2}\right) - (k/\sqrt{n(k)})$ 。

E.1.3 两个计算问题

本节简要地回顾两个放宽后的传统编码理论任务。放宽的目的是使有意义的超高速 (随机) 算法成为可能。具体地说, 这些算法可能运行在亚线性时间 (如多项式对数), 因此不可能解决未放宽版本的问题。

局部可测试性: 这个任务是指基于词中 (随机) 检查的几个位置, 测试一个给定的字是否为码字 (在预定的码中)。不用说, 只能寄希望于一个近似正确的判定; 也就是说, 接受每个码字, 且以高的概率拒绝距码很远的词 (事实上, 这个任务是在属性测试框架内的)。

局部可译码性: 这个任务是指从轻微受损的码字中几个特定的 (随机的) 检查位置来恢复明文的特定位置。与这项任务有一定关系的是自纠错 (即通过检查轻度损坏的码字的几个特定位置, 恢复出码字本身的特定位置。)

注意: Hadamard 码是局部可测试的, 也是局部可译以及自纠错的 (通过在字中进行固定数目的询问)。这些事实都已被证实, 并在 9.3.2.1 节广泛使用。然而, Hadamard 码具有指数级的分组长度 (即 $n(k) = 2^k$), 问题是是否可以实现相对于较短的 (如 $n(k) = \text{poly}(k)$) 类似的码。附录 E.1.1.4 中暗示了答案是肯定的 (当在 k 的多项式对数上执行这些操作时)。

定理 E.3 对于常量 $\delta > 0$ 和多项式 $n, q: \mathbf{N} \rightarrow \mathbf{N}$, 存在相对距离为 δ 的明确性的码族 $\{C_k: [q(k)]^k \rightarrow [q(k)]^{n(k)}\}_{k \in \mathbf{N}}$, 能够在 $\text{poly}(\log k)$ 时间内是局部可测试和局部可译码。即能够满足如下 3 个条件:

(1) 编码。存在多项式时间算法 m , 对输入 $x \in [q(k)]^k$ 返回 $C_k(x)$ 。

(2) 局部测试。存在概率多项式时间的预言机 T , 给定 k (二进制表示) ^①, 对预言机访问 $w \in [q(k)]^{n(k)}$, 区分 w 是码字和 w 与任意码字 $\delta/2$ 远的情况。

① 对每个 $x \in [q(k)]^k$ 有 $\Pr[T^{C_k(x)}(k) = 1] = 1$ 。

② 对距离 C_k 的任意码字 $\delta/2$ 远的每个 $w \in [q(k)]^{n(k)}$, 有 $\Pr[T^w(k) = 1] \leq 1/2$ 。

照例可通过重复来降低错误概率。

(3) 局部译码。存在概率多项式时间的预言机 D , 为其给定 k 和 $i \in [k]$ (二进制形式), 对任意距离 $C_k(x)$ $\delta/2$ 近的预言机访问 $w \in [q(k)]^{n(k)}$ 返回 x_i , 即 $\Pr[D^w(k, i) = x_i] \geq 2/3$ 。

除此之外, 还具有自纠错性质, 即存在一个概率多项式时间预言机 M , 给定 k 和 $i \in [n(k)]$ (以二进制形式), 以及对任意与 $C_k(x)$ $\delta/2$ 近的预言访问 $w \in [q(k)]^{n(k)}$, M 返回 $C_k(x)$ 。也就是说, $\Pr[D^w(k, i) = C_k(x)_i] \geq 2/3$ 。

要强调的是, 所有这些预言机工作于 k 的二进制表示的多项式时间上, 这意味着, 它们

① 从而 T 的运行时间为 $\text{poly}(|k|) = \text{poly}(\log k)$ 。

工作于 k 的多项式对数时间上。定理 E.3 的码可认定是（稍做修改的）Reed-Muller 码，其中 $r = m^2 \log m < q(k) = \text{plog}(r)$ ， $[n(k)] \equiv \text{GF}(q(k))^m$ （见附录 E.1.2.4）。^①上述预言机对非固定数量的位置进行预言询问 $w: [n(k)] \rightarrow \text{GF}(q(k))$ 。具体来说，对位置 $i \in \text{GF}(q(k))^m$ 的自纠错是通过随机选择一个能通过 i 的行，作为 w 到所有在这个行上 $q(k)$ 点的符号来恢复值，并进行单变量多项式推广（在低噪声下）。局部可测试性容易归约到自纠错，并且（根据上述的变形例）局部可译码性是自纠错的一种特例。

恒定数量的查询

在定理 E.3 中断言的局部测试和译码算法对预言机进行询问的次数是多项式的。相比之下，Hadamard 码能以恒定数量的查询来支持这些操作。能否以更短的码字来获得这些操作呢？对于局部可测试性的回答是肯定的。可以得到接近线性长度的局部可测试码（即从线性到多项式对数，参见参考文献[36, 67]）。对基于恒定数量查询的局部可译码性，已知最短的码也具有超多项式的长度（见参考文献[242]）。鉴于这种状况，主张将局部可译码任务放宽（如参考文献[35]中的情况）。

有兴趣的读者可以见参考文献[93]，其中包括局部可测试和可译码的详细信息（但是请注意，本综述先于参考文献[67]和[242]，它解决了参考文献[93]中讨论的两个主要的公开问题）。

E.1.4 列表译码的上界

对于列表译码任务的可行性，一个必要条件是接近给定字的码字列表很短。本节对这个列表的长度给出上界，并指出这个界在复杂性理论的很多场合（和本书中的相关特定内）都有所涉及。相比之下，不存在更著名的界（这通常涉及码的几个主要参数（即 k 、 n 和 d ）间的关系），因为它们看起来与本书的内容无关。

首先对任何字母 $\Sigma \equiv [q]$ 作简要声明，然后具体讨论 $q = 2$ 的情况。特别是在一般情况下，考虑 $[q]$ 上序列之间的一致性（而不是距离）是很方便的。此外，集中在一致率（Agreement Rate）至少在 $1/q$ 上是自然的，这也可以说明“过度一致率”（即超过 $1/q$ ）的如下结果。^②

引理 E.4（参考文献[105]中的定理 15 的第 2 部分） 令 $C: [q]^k \rightarrow [q]^n$ 距离为 $d \leq n \cdot (n/q)$ 的任意码， $\eta_c \stackrel{\text{def}}{=} (1 - (d/n)) - (1/q) \geq 0$ 表示码字间超出一致性的相应上界。假定 $\eta \in \{0, 1\}$ 满足

$$\eta > \sqrt{1 - \frac{1}{q} \eta_c} \quad (\text{E.2})$$

于是，对于任意 $w \in [q]^n$ ，与 w 至少有个 $((1/q) + \eta) \cdot n$ 位置一致（即与 w 的距离致多为 $(1 - ((1/q) + \eta)) \cdot n$ ）的码字数量上界为

$$\frac{(1 - (1/q))^2 - (1 - 1/q) \cdot \eta_c}{\eta^2 - (1 - (1/q)) \cdot \eta_c} \quad (\text{E.3})$$

在二进制情况下（即 $q=2$ ），式 (E.1) 要求 $\eta > \sqrt{\eta_c/2}$ 且式 (E.2) 产生上界 $(1 - 2\eta_c) / (4\eta^2 - 2\eta_c)$ 。要强调两种特殊情况：

- ① 这种修改类似于脚注：对适当选择的 k 个点 $\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_k \in \text{GF}(q(k))^m$ ，将 $(v_1, v_2, \dots, v_k)$ 映射到 $(p(\bar{\alpha}_1), p(\bar{\alpha}_2), \dots, p(\bar{\alpha}_k))$ ，其中 p 为次数不超过 r 的唯一的 m -变量多项式，且满足 $p(\bar{\alpha}_i) = v_i$ ， $i = 1, 2, \dots, k$ 。
- ② 事实上，我们只考虑距离为 $d \leq (1 - 1/q) \cdot n$ 的码（即一致率至少为 $1/q$ ）和与码字距离不超过 d 的字。注意：一个随机序列与任意固定序列至少在 $1/q$ 的位置是一致的。

(1) 在附录 D.4.4.4 结尾, 涉及到了这个界 (对二进制的情况), 此时, $\eta_c = (1/k)^2$ 且 $\eta = 1/k$ 。实际上, 在此情况下, $(1-2\eta_c)/(4\eta^2-2\eta_c) = O(k^2)$ 。

(2) 在 Hadamard 码中, 有 $\eta_c = 0$ 。于是, 对所有 $w \in \{0, 1\}^n$ 和所有 $\eta > 0$, $(0.5-\eta)$ 接近于 w 的码字数至多为 $1/(4\eta^2)$ 。

在一般情况下 (特别是 $q \gg 2$), 通过 $\eta > \min\{\sqrt{\eta_c}, \left(\frac{1}{q}\right) + \sqrt{\eta_c - (1/q)}\}$ 来简化式 (E.2), 通过 $\frac{1}{\eta^2 - \eta_c}$ 来简化式 (E.3) 是很有用的。

E.2 扩张图

本节回顾与本书内容相关的扩张图的基本结论。感兴趣的读者可进一步见参考文献[124]。

粗略地说, 扩张图是度数较小的正则图, 其具有团的各种性质。^①特别是, 本节涉及到的性质包括图中切割的相对规模, 以及随机漫游收敛于均匀分布 (与以图的度为底, 图的规模的对数相关) 的比率。

一些技术细节

扩张图的典型表示方法有几种变形。如在一些资料中, 扩张图被表示为二分图, 而在其他资料中可能被表示为普通图 (并且与二分图相去甚远)。本文遵循后一种习惯。此外, 有时隐含地考虑这样的图, 其中为每个顶点增加一个自环。为简单起见, 也允许平行边。

通常所说的扩张图是指满足相同属性 (被非正式地归于集合) 的图的无限集。例如, 在谈及一个 d -正则扩展 (图) 时, 实际上是这些 d -正则图的无限集合。一般情况下, 这样的集合 (或族) 对任意 $N \in S$, 包含一个 N 顶点图, 其中 S 是自然数集的无限子集。本节用 $\{G_N\}_{N \in S}$ 表示该集合, 其中 G_N 为含 N 个顶点的图。

E.2.1 定义和性质

考虑扩张图的两种定义、两种明确构造以及两个有用的属性。

E.2.1.1 两种数学定义

首先介绍扩张图的两种不同定义。这些定义本质上等价, 甚至定量考虑时也是相关的。先从一个代数定义开始, 这实际上是复杂性理论中的典型定义, 因为它直接蕴含着不同的“混合属性”。后面提出了一个非常自然的组合定义 (也就是术语“扩张”的来源)。

代数定义 (特征值缝隙)

用图的邻接矩阵识别一个图, 考虑图 (或者确切地说是其邻接矩阵) 的特征值 (和特征向量)。任何 d -正则图 $G=(V,E)$, 具有均匀的向量作为对应于特征值 d 的特征向量, 并且如果 G 是连通的而不是二分的, 则所有其他特征值 (的绝对值) 严格小于 d 。这样的图 G 的第二个特征值, 记为 $\lambda_2(G) < d$, 是所有其他特征值的绝对值的紧致上界。使用与组合相关的定义,

① 另一种有用的直观感觉是扩张图具有相同度数随机正则图的各种性质。

有 $\lambda_2(G) < d - \Omega(1/d|V|^2)$ 成立(对于所有连通的非二分 d -正则图 G)。^①扩张的代数定义是指 d -正则图的无限族, 需要该族中的所有图保持恒定的特征值。

定义 E.5 如果对所有 $N \in S$ 有 $\lambda(G_N) \leq \beta$ 成立, 则称一个 d -正则图的无限族 $\{G_N\}_{N \in S}$, 满足特征值界 β , 其中 $S \subseteq \mathbf{N}$ 。此时, 称 $\{G_N\}_{N \in S}$ 是一个 (d, β) -扩张图族, $d - \beta$ 为其特征值缝隙。

组合定义(扩张)

粗略地说, 扩张要求任何图的顶点集(不是太大)具有相对较大的邻居集合。具体地, 一个图 $G=(V, E)$, 对于至多含 $|V|/2$ 个元素的任意集合 $S \subset V$, 如果满足

$$\Gamma_G(S) \stackrel{\text{def}}{=} \{v: \exists u \in S \text{ s.t. } \{u, v\} \in E\} \quad (\text{E.4})$$

至少有 $(1+c) \cdot |S|$ 个元素, 则称 G 是一个 c -扩张图。假设图中存在自环, 则上述要求等价于 $|\Gamma_G(S) \setminus S| \geq c \cdot |S|$ 。显然, 每一个连通图 $G=(V, E)$ 为 $(1/|V|)$ -扩张图。^②扩张的组合定义是指 d -正则图的无限族, 且族中所有图都存在常数的扩张界。

定义 E.6 如果对所有 $N \in S$, G_N 是 c -扩张的, 则一个 d -正则图的无限族 $\{G_N\}_{N \in S}$ 是 c -扩张的。

扩张图的两个定义是相关的(见参考文献[11]中的 9.2 节或参考文献[124]中的 4.5 节)。具体地, “扩张界限”与“特征值界限”间的关系如下。

定理 E.7 令 G 是每个顶点均有自环的 d -正则图, ^③则

(1) 对于 $c \geq (d - \lambda_2(G))/2d$, G 是 c -扩张的。

(2) 如果 G 是 c -扩张的, 则 $d - \lambda_2(G) \geq c^2/(4+2c^2)$ 。

因此, d -正则图族的组合扩张的任何非零界, 会产生其特征值缝隙的一个非零界, 反之亦然。但是请注意, 这些定义之间的前后转换不严密。本书正文中(见 8.5.3 节和 9.3.2.3 节)主要指的是代数定义, 定理 E.7 引入的损失并不重要。

放大

“扩张图的质量改善”可通过将其提高到任意幂 $t > 1$ (即将其邻接矩阵提高到 t 次幂), 在对应的图中是指将(原始图中的) t 路径用边来代替。应用代数定义, 有 $\lambda(G^t) = \lambda(G)^t$, 但是实际上也被提高到 t 次幂。比率 $\lambda(G^t)/d^t$ 随 t 的增大而降低。在组合定义中也有类似现象, 条件是要进行适当的修改。例如, 如果 $G=(V, E)$ 是 c -扩张的, 那么, 对于每个 $S \subseteq V$, 有 $|\Gamma_G(S)| \geq \min((1+c) \cdot |S|, |V|/2)$ 。

最优特征值界

对于每个 d -正则图 $G=(V, E)$, 有 $\lambda(G) \geq 2\gamma_G \cdot \sqrt{d-1}$, 其中 $\gamma_G = 1 - O(1/\log_d |V|)$ 。于是,

① 这可由与组合定义(见定理 E.7)的关系得到。具体地, 图的平方, 记作 G^2 , 是 $|V|^{-1}$ 扩张的, 从而有 $\lambda(G)^2 = \lambda(G^2) < d^2 - \Omega(|V|^2)$ 。

② 相反, 一个二分图 $G=(V, E)$ 并不是扩张的, 因为其中总包含规模至多为 $|V|/2$ 的集合 S , 满足 $|\Gamma_G(S)| \leq |S|$ (虽然可能有 $|\Gamma_G(S) \setminus S| \geq |S|$)。

③ 回顾在这样一个图 $G=(V, E)$ 中, 对每个 $S \subseteq V$, 有 $\Gamma_G(S) \supset S$, 从而 $|\Gamma_G(S)| = |\Gamma_G(S) \setminus S| + |S|$ 。进一步地, 在该图中, 所有特征值大于或等于 $-d+1$, 从而如果 $d - \lambda(G) < 1$, 则这是由于 G 的正特征值所致。这些事实可用于桥接附录 E.7 与更标准版本之间的缝隙(见参考文献[11]中的 9.2 节)。具体地, 参考文献[11]中的 9.2 节针对着 $\Gamma_G^+(S) = \Gamma_G(S) \setminus S$ 及 $\lambda_2(G)$, 其中 $\lambda_2(G)$ 是 G 的次大特征值, 而非针对 $\Gamma_G(S)$ 和 $\lambda(G)$ 。注意: 在一般情况下, $\Gamma_G(S)$ 可能位于 G 的最小特征值与 $-d$ 之间。

对任意 (d, λ) -扩张图, 必有 $\lambda \geq 2\sqrt{d-1}$ 。

E.2.1.2 明确性的两个层次

在讨论图的明确构造的各种概念时, 我们需要指定图的表达方式。具体地, 在本节, 当提到有限的图集 $\{G_N\}_{N \in S}$ 时, 总是假设 G_N 的顶点集等于 $[N]$ 。事实上, 有时会考虑具有不同结构的顶点集 (如对某个 $m \in \mathbb{N}$, 顶点集为 $[m] \times [m]$), 但是在所有情况中, 这些集合的规范表示间存在一种简单的同构关系 (即在 G_N 的顶点集到 $[N]$ 存在一种可高效计算且可逆的映射)。

明确性构造的温和层次指的是构造整个对象 (即图) 的复杂性。因此, 如果存在一个多项式时间算法, 对输入 1^N , 输出 N 个顶点的图形 G_N 的边, 则图 $\{G_N\}_{N \in S}$ 的无限族是可明确构造的。

当应用程序需要在整个图中运行, 且/或运行时间是图的规模下界时, 上述的明确性程度足够了。相比之下, 其他应用仅涉及一个巨大的虚拟图 (比其运行时间更大), 并且只需要计算图中的邻居关系。在此情况下, 需要如下更强层次的明确性。

(d -正则) 图 $\{G_N\}_{N \in S}$ 的无限族更强的明确构造是一个多项式时间算法, 对输入的 N (二进制)、 N 顶点图 G_N 的顶点 v 和索引 i ($i \in \{1, 2, \dots, d\}$), 返回 v 的第 i 个邻居。即在图的规模的多项式-对数时间内返回对其“邻居结点询问”的应答。当然, 更强明确性的层次蕴含了基本层次。

经常被遗忘但很重要的一个附加要求是集合 S 的“易处理性”。具体而言, 要求存在一个高效算法, 对给定 $n \in \mathbb{N}$ 查找 $s \in S$ 使得 $n \leq s < 2n$ 。对应于上述定义, 高效可能意味着在时间 $\text{poly}(n)$ 上运行或是在时间 $\text{poly}(\log n)$ 上运行。 $n \leq s < 2n$ 的要求对于大多数应用是充分的, 但在有些情况下必须有较小的间隔 (如 $n \leq s < n + \sqrt{n}$), 而在其他情况下使用较大的间隔 (如 $n \leq s < \text{poly}(n)$) 就足够了。

更强的灵活性

续前一段, 可以扩张图进行组合以得到更大规模的扩张图。例如, 两个 d -正则、 c -扩张的图 $G_1 = (V_1, E_1)$ 和 $G_2 = (V_2, E_2)$ ($|V_1| \leq |V_2|, c \leq 1$), 可以组合为一个 $(d+1)$ -正则、 $c/2$ -扩张的含 $|V_1| + |V_2|$ 个结点的图, 方法是利用 V_1 与 V_2 中 $|V_1|$ 个顶点之间的完美匹配连接两个图 (并对 V_2 中剩余的顶点添加自环)。更普遍地, 将 d -正则、 c -扩张的图 $G_1 = (V_1, E_1)$ 与 $G_t = (V_t, E_t)$ 组合起来, 其中 $N' \stackrel{\text{def}}{=} \sum_{i=1}^t |V_i| \leq |V_t|$, 可得到一个 $(d+1)$ -正则、 $c/2$ -扩张的图, 其中含 $\sum_{i=1}^t |V_i|$ 个顶点 (利用 $\bigcup_{i=1}^t V_i$ 与 V_t 中 N' 个顶点间的完美匹配)。

E.2.1.3 两个属性

以下两个属性提供了定量的解释来说明扩张图近似为完全图。如果是 (d, λ) -扩张图, 则其与完全图的偏差可由一个与 λd 呈线性关系的误差项来表示。

混合引理

粗略地说, 以下引理断言在扩张图 (其中 $\lambda \ll d$) 中, 连接两个较大顶点集合的边的比例等于这些集合密度的乘积。这一属性称为混合。

引理 E.8 (扩张混合引理) 对于所有 d -正则图 $G=(V, E)$ 以及两个子集 $A, B \subseteq V$, 有如下关系成立, 即

$$\left| \frac{|(A \times B) \cap E|}{|E|} - \frac{|A|}{|V|} \cdot \frac{|B|}{|V|} \right| \leq \frac{\lambda(G) \sqrt{|A| \cdot |B|}}{d \cdot |V|} \leq \frac{\lambda(G)}{d} \quad (\text{E.5})$$

式中: E 表示对应于 G 的无向边 (即顶点对) 的有向边的集合 (即 $E = \{(u, v) : \{u, v\} \in E\}$ 且 $|E| = d|V|$)。

证明: 令 $N = |V|$, $\lambda = \lambda(G)$ 。对顶点的任意子集 $S \subseteq V$, 用 $\rho(S) = |S|/N$ 表示其在 V 中的密度, 则式(E.5)可重新表示为

$$\left| \frac{|(A \times B) \cap E|}{d \cdot N} - \rho(A) \cdot \rho(B) \right| \leq \frac{\lambda \sqrt{\rho(A) \cdot \rho(B)}}{d}$$

继续给出 $(A \times B) \cap E$ 取值的上界。令 $\mathbf{a}$ 表示 N 维布尔向量, 当且仅当 $i \in A$ 时其第 i 个分量为 1。向量 $\mathbf{b}$ 的定义类似。用 $\mathbf{M} = (m_{i,j})$ 表示图 G 的邻接矩阵, 则 $(A \times B) \cap E$ 等于 $\mathbf{a}^T \mathbf{M} \mathbf{b}$ (当且仅当 $i \in A$, $j \in B$ 且 $m_{i,j} = 1$ 时, $(i, j) \in (A \times B) \cap E$)。考虑正交的特征向量基 $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \dots, \bar{\mathbf{e}}_N$, 其中 $\bar{\mathbf{e}}_1 = (1, 1, \dots, 1)^T$, 且对每个 i 有且 $\bar{\mathbf{e}}_i^T \bar{\mathbf{e}}_i = N$, 将每个向量记作该基中向量的线性组合。具体而言, 用 a_i 表示 $\mathbf{a}$ 在 $\bar{\mathbf{e}}_i$ 方向上的系数, 即 $a_i = (\mathbf{a}^T \bar{\mathbf{e}}_i) / N$ 且 $\mathbf{a} = \sum_i a_i \bar{\mathbf{e}}_i$ 。注意: $a_1 = (\mathbf{a}^T \bar{\mathbf{e}}_1) / N = |A| / N = \rho(A)$ 且 $\sum_{i=1}^N a_i^2 = (\mathbf{a}^T \mathbf{a}) / N = |A| / N = \rho(A)$ 。对 $\mathbf{b}$ 也有类似情况。于是, 有如下关系成立, 即

$$|(A \times B) \cap E| = \mathbf{a}^T \mathbf{M} \sum_{i=1}^N b_i \bar{\mathbf{e}}_i = \sum_{i=1}^N b_i \lambda_i \cdot \mathbf{a}^T \bar{\mathbf{e}}_i$$

式中: λ_i 表示 $\mathbf{M}$ 的第 i 个特征值, 注意: $\lambda_1 = d$ 且对任意 $i \geq 2$, 有 $|\lambda_i| \leq \lambda$, 从而

$$\left| \frac{|(A \times B) \cap E|}{d \cdot N} \right| = \sum_{i=1}^N \frac{b_i \lambda_i \cdot a_i}{d} = \rho(A) \cdot \rho(B) + \sum_{i=2}^N \frac{\lambda_i a_i b_i}{d} \in \left[\rho(A) \cdot \rho(B) \pm \frac{\lambda}{d} \cdot \sum_{i=2}^N a_i b_i \right]$$

由于 $\sum_{i=1}^N a_i^2 = \rho(A)$, $\sum_{i=1}^N b_i^2 = \rho(B)$, 并应用 Cauchy-Schwartz 不等式, 可由 $\sqrt{\rho(B)\rho(A)}$ 界定 $\sum_{i=2}^N a_i b_i$ 。引理得证。 ■

随机漫游引理

粗略地讲, 下面引理的第一部分称, 只要顶点集保持“陷入”在某个子集之内, 则扩张图上的随机漫游接近于完全图上的随机漫游。

引理 E.9 (扩张图上的随机漫游引理) 设 $G = ([N], E)$ 是一个 d -正则图, 从均匀选择的顶点开始 G 上漫游, 并多走 $l-1$ 步, 在每一步, 从当前顶点依附的 d 条边中均匀选择一条并通过。

定理 8.28 (重述) 令 W 为 $[N]$ 的子集, 且 $\rho = |W|/N$, 则这样的随机漫游停留在 W 中的概率至多为

$$\rho \cdot \left(\rho + (1-\rho) \cdot \frac{\lambda(G)}{d} \right)^{l-1} \quad (\text{E.6})$$

习题 8.43 (重述) 对于任意 $W_0, W_1, \dots, W_{l-1} \subseteq [N]$, 长度为 l 的随机漫游贯穿 $W_0 \times W_1 \times \dots \times W_{l-1}$ 的概率至多为

$$\sqrt{\rho_0} \cdot \prod_{i=1}^{l-1} \sqrt{\rho_i + (\lambda/d)^2} \quad (\text{E.7})$$

其中

$$\rho_i \stackrel{\text{def}}{=} |W_i|/N$$

引理 E.9 依据的基本原理是由 Ajtai、Komlos 和 Szemerédi^[4]发现的, 他们证明了式(E.7) 中的界。由 Kahale (参考文献[135]中的推论 6.1) 给出的更好的分析产生了定理 8.28。参考文献[120]给出了关于以大约 $|W|/N$ 的次数访问 W 的概率的更一般界, 这实际上针对更一般的问题 (即获得由一个扩张图上的随机漫游生成的随机变量的切诺夫型界)。

公式 (E.6) 证明: 基本思路是, 将随机漫游中发生的事件视为相应概率向量在适当变换下的演变。该转换与取自 G 中的随机步骤相对应, 并通过一个“筛子”, 仅保留对应于当前集合 W_i 中的项。关键在于, 第一个变换使得与均匀分布正交的分量收缩, 而第二个变换使与均匀分布方向相同的分量收缩。具体情况如下:

设矩阵 A 为 G 上的随机漫游 (即 A 是 G 的邻接矩阵除以 d), 并用 $\hat{\lambda} \stackrel{\text{def}}{=} \lambda(G)/d$ (即 $\hat{\lambda}$ 为 A 的特征值的绝对值上界 (第一个除外))。需要注意, 由向量 $\bar{u} = (N^{-1}, \dots, N^{-1})^T$ 表示的均匀分布, 是与 A 的最大特征值有关的特征向量。设 P_l 为仅在对角线取 1 的 0-1 矩阵, 进而当且仅当 $j \in W_l$ 时向量 (j, j) 设定为 1。于是, 长为 l 的随机漫游距离的贯穿 $W_0 \times W_1 \times \dots \times W_{l-1}$ 的概率为向量

$$\bar{v} \stackrel{\text{def}}{=} P_{l-1} A \cdots P_2 A P_1 A P_0 \bar{u} \quad (\text{E.8})$$

中各项之和。

此处, 我们对上界 $\|\bar{v}\|$ 感兴趣, 并利用 $\|\bar{v}\| \leq \sqrt{N} \cdot \|v\|$ 其中 $\|\bar{z}\|$ 和 $\|z\|$ 分别表示 $\bar{z}$ 的 L_1 -范数和 L_2 -范数 (如 $\|\bar{u}\|_1 = 1$ 和 $\|\bar{u}\|_2 = N^{-1/2}$)。关键是线性变换 $P_l A$ 使每个向量收缩。

主要声明

对于任意 $\bar{z}$, 有 $\|P_l A \bar{z}\| \leq (\rho_l + \hat{\lambda}^2)^{1/2} \cdot \|\bar{z}\|$ 成立。

证明: 直观地看, A 使 $\bar{z}$ 中与 $\bar{u}$ 正交的分量收缩, 而 P_l 使 $\bar{z}$ 中与 $\bar{u}$ 方向相同的分量收缩。特别地, 分解 $\bar{z} = \bar{z}_1 + \bar{z}_2$, 使得 $\bar{z}_1$ 是 $\bar{z}$ 在 $\bar{u}$ 上的投影, 而 $\bar{z}_2$ 是与 $\bar{u}$ 正交的分量。然后, 利用三角不等式和其他一些事实 (其蕴着 $\|P_l A \bar{z}_1\| = \|P_l \bar{z}_1\|$ 和 $\|P_l A \bar{z}_2\| \leq \|A \bar{z}_2\|$), 可以得到

$$\|P_l A \bar{z}_1 + P_l A \bar{z}_2\| \leq \|P_l A \bar{z}_1\| + \|P_l A \bar{z}_2\| \leq \|P_l \bar{z}_1\| + \|A \bar{z}_2\| \leq \sqrt{\rho_l} \cdot \|\bar{z}_1\| + \hat{\lambda} \cdot \|\bar{z}_2\|$$

其中最后一个不等式是由于 P_l 在收缩任意均匀向量时, 消除了其中 $1 - \rho_l$ 个分量。而 A 以至少 $\hat{\lambda}$ 的因子使除 $\bar{u}$ 之外任意特征向量的长度收缩。由 Cauchy-Schwartz 不等式^①, 得到

$$\|P_l A \bar{z}_1\| \leq \sqrt{\rho_l + \hat{\lambda}^2} \cdot \sqrt{\|\bar{z}_1\|^2 + \|\bar{z}_2\|^2} = \sqrt{\rho_l + \hat{\lambda}^2} \cdot \|\bar{z}\|$$

其中等式成立是因为 $\bar{z}_1$ 与 $\bar{z}_2$ 正交。 □

回顾式 (E.8), 并应用主要结论 (及 $\|\bar{v}\| \leq \sqrt{N} \cdot \|v\|$), 得到

① 利用 $\sum_{i=1}^n a_i \cdot b_i \leq (\sum_{i=1}^n a_i^2)^{1/2} \cdot (\sum_{i=1}^n b_i^2)^{1/2}$, 并令 $n=2, a_1 = \sqrt{\rho_l}, b_1 = \|\bar{z}_1\|$, 可得到 $\sqrt{\rho_l} \cdot \|\bar{z}_1\| + \hat{\lambda} \cdot \|\bar{z}_2\| \leq \sqrt{\rho_l + \hat{\lambda}^2} \cdot \sqrt{\|\bar{z}_1\|^2 + \|\bar{z}_2\|^2}$ 。

$$\|\bar{v}\|_1 \leq \sqrt{N} \cdot \|P_{\ell-1} A \cdots P_2 A P_1 A P_0 \bar{u}\| \leq \sqrt{N} \cdot \left(\prod_{i=1}^{\ell-1} \sqrt{\rho_i + \hat{\lambda}^2} \right) \cdot \|P_0 \bar{u}\|$$

最后, 由 $\|P_0 \bar{u}\| = \sqrt{\rho_0 N \cdot (1/N)^2} = \sqrt{\rho_0 / N}$, 得到式 (E.7)。

快速混合

与引理 E.9 相关的一个性质是从任意顶点开始的随机漫游在经过对数步后, 收敛于扩张图上顶点的均匀分布。具体地, 在一个 (d, λ) -扩张图 $G = ([N], E)$ 上, 从任意分布 $\bar{s}$ (包括将所有权重分配给一个顶点的情况) 开始, 经过 l 步后, 最终到达的分布与 $[N]$ 上的均匀分布是 $\sqrt{N} \cdot (\lambda/d)^l$ -接近的。利用式 (E.7) 证明中的符号, 该断言称仅当 $l > 0.5 \cdot \log_{1/\hat{\lambda}} N$ 时, $\|A^l \bar{s} - \bar{u}\| \cdot \sqrt{N} \cdot \hat{\lambda}^l$ 有意义。注意到, $\|A^l \bar{s} - \bar{u}\| \leq \sqrt{N} \cdot \|A^l \bar{s} - \bar{u}\|$, 并利用 $\bar{s} - \bar{u}$ 与 $\bar{u}$ 正交的事实可证明该断言 (因为前者的各项之和为零), 从而有 $\|A^l \bar{s} - \bar{u}\| = \|A^l (\bar{s} - \bar{u})\| \leq \hat{\lambda}^l \|\bar{s} - \bar{u}\|$, 应用 $\|\bar{s} - \bar{u}\| < 1$, 断言得证。

E.2.2 构造

已经发现了许多 (d, λ) -扩张图的明确构造, 第一个构造出现于参考文献[164], 并最终形成了参考文献[160]中的优化结构, 其中 $\lambda = 2\sqrt{d-1}$ 。大多数构造是相当简单的 (见附录 E.2.2.1), 但对其分析利用了来自各个数学分支中的先进结果。相比之下, Reingold、Vadhan 和 Wigderson^[191]在附录 E.2.2.1 提出的构造是基于一个迭代过程, 其分析利用的则是与矩阵特征值有关的相对简单的代数知识。

在讨论这些明确性构造之前, 注意到使用概率方法 (参见参考文献[11]) 并参照扩张图的组合定义, 比较容易证明 3-正则扩张图的存在性。

E.2.2.1 Margulis–Gabber–Galil 扩张图

对任意自然数 m , 考虑顶点集合为 $\mathbf{Z}_m \times \mathbf{Z}_m$ 的图, 其中每条边 $\langle x, y \rangle \in \mathbf{Z}_m \times \mathbf{Z}_m$ 关联于顶点 $\langle x \pm y, y \rangle$ 、 $\langle x \pm (y+1), y \rangle$ 和 $\langle x, y \pm x \rangle$, 此处的运算要模 m 。这就产生了一个非常简单且明确的 8-正则图, 其特征值上限是独立于 m 的常量 $\lambda < 8$ 。由此得到:

定理 E.10 存在规模为 $\{m^2: m \in \mathbf{N}\}$ 的 $(8, 7.9999)$ -扩张图族的强明确构造。而且, 图中顶点的邻居能在对数空间内计算求得。^①

定理 E.10 的一个吸引人的特性是, 对任意 $n \in \mathbf{N}$, 它直接利用顶点集 $\{0, 1\}^n$ 产生扩张图。当 n 为偶数时这很显然, 但 n 为奇数时也很容易实现 (如对于 $n-1$ 使用该图的两个副本, 并以明显的完美匹配连接这两个副本)。

定理 E.10 是由 Gabber 和 Galil^[84]提出的, 并基于 Margulis^[164]提出的基本方法。我们再次指出参考文献[160]中的最优结构可以实现 (即 $\lambda = 2\sqrt{d-1}$), 但对度数 d 有一些限制 (即 $d-1$ 为模 4 余 1 的素数), 且对图的规模也有限制。^②

E.2.2.2 迭代 Zig-Zag 构造

以下构造的出发点是具有常数规模的非常好的扩张图 G , 该图可以通过穷举搜索找到。

① 事实上, 当 m 为 2 的幂时 (在适当的顶点编码下), 可利用运行于常数空间的在线算法计算邻居。将顶点 $\langle x, y \rangle$ 与顶点 $\langle x \pm 2y, y \rangle$ 、 $\langle x \pm (2y+1), y \rangle$ 、 $\langle x, y \pm 2x \rangle$ 及 $\langle x, y \pm (2x+1) \rangle$ 相关联时, 也可这样计算, 这种变形可得到更好的 λ 上限, 即 $\lambda \leq 5\sqrt{2} \approx 7.071$ 。

② 参考文献[160]中的构造允许形如 $(p^3 - p)/2$ 的图规模, 其中 $p \equiv 1 \pmod{4}$ 为素数, 且 $d-1$ 是模 p 的二次剩余。如参考文献[8]中的第 2 节所述, 该构造可扩展到规模为 $(p^{3k} - p^{3k-2})/2$ 的图, 其中 $k \in \mathbf{N}$, p 同上。

为得到一个较大的扩张图, 采用迭代方法, 其中在 i 次迭代时将当前图 G_i 与给定图 G 组合, 产生一个大图 G_{i+1} 。组合步骤保证了 G_{i+1} 的扩张特性至少和 G_i 一样好, 而 G_{i+1} 保持了 G_i 的度且比 G_i 大了常数倍。该过程开始于 $G_1=G^2$, 并在得到了大致等于所需规模的图 G_i 时终止 (需要进行对数次迭代)。

Zig-Zag 积

组合步骤的核心是一种称为 zig-zag 积的新型“图积”。这种操作适用于任意一对图 $G=([D], E)$ 和 $G'=([N], E')$, 条件是 G' (通常大于 G) 是 D -正则的。为简化起见, 假定 G 是 d -正则的 (典型地, $d \ll D$)。 G' 与 G 的 zig-zag 积, 记作 $G' \otimes G$, 是一个顶点集为 $[N] \times [D]$ 的图, 其边集中包含边 $\langle u, i \rangle \in [N] \times [D]$ 和 $\langle v, j \rangle$ 当且仅当 $(i, k), (k, j) \in E$, 且依附于 u 的第 k 条边等于依附于 v 的第 l 条边。即 $\langle u, i \rangle$ 和 $\langle v, j \rangle$ 在 $G' \otimes G$ 中相连的条件是存在如下的“3 步序列”: 在 G 中由 $\langle u, i \rangle$ 到 $\langle u, k \rangle$ (对应于 G 中的边 $\{i, k\}$), 在 G' 中由 $\langle u, k \rangle$ 到 $\langle v, l \rangle$ (对应于 u 在 G' 中的第 k 条边 (是 v 的第 l 条边)), 再在 G 中 $\langle v, l \rangle$ 由到 $\langle v, j \rangle$ (对应于 G 中的边 $\{l, j\}$)。参见图 E.1, 进一步说明如下。

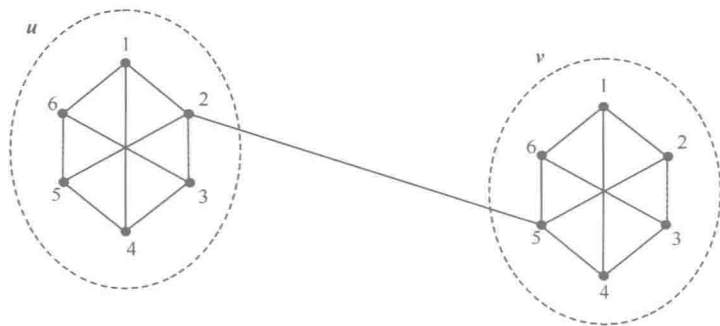


图 E.1 G 和 G' 的 zig-zag 积细节。在这个例子中, G' 是 6-正则图, G 是有 6 个顶点的 3-正则图。在图 G' 中 (未显示), 顶点 u 上的第二个边是从 v 接入的, 作为第五个边。3-段线表示 $G' \otimes G$ 的对应边,

该边连接了 $\langle u, 3 \rangle$ 和 $\langle v, 2 \rangle$

教学注释: 以下段落给出了 zig-zag 积的正式描述, 在快速阅读时可以忽略, 但对更进一步的讨论非常有用。

通过记作 $R': [N] \times [D] \rightarrow [N] \times [D]$ 的边旋转函数来表示像 G' 一样的图是很方便的, 如果 (u, v) 是依附于 u 的第 i 条边, 也是依附于 v 的第 j 条边, 那么, $R'(u, i) = (v, j)$ 。为简便起见, 假设常数规模 d -正则图 $G=([D], E)$ 可以利用 d 种颜色进行边着色, 这反过来又产生了自然的边旋转函数 (即如果边 $\{i, j\}$ 被着色为 α , 则 $R(i, \alpha) = (j, \alpha)$)。用 $E_\alpha(i)$ 表示从 $i \in [D]$ 出发, 经过被着色为 α 的边后到达的顶点 (即若 $R(i, \alpha) = (j, \alpha)$, 则 $E_\alpha(i) = j$)。 G' 和 G 的 zig-zag 积, 记作 $G' \otimes G$, 定义为顶点集为 $[N] \times [D]$ 的图, 边旋转函数定义为

$$\text{如果 } R'(u, E_\alpha(i)) = (v, E_\beta(j)) \text{ 则 } (\langle u, i \rangle, \langle \alpha, \beta \rangle) \rightarrow (\langle u, j \rangle, \langle \beta, \alpha \rangle) \quad (\text{E.9})$$

换言之, 边由 $[d]$ 上的对进行标记。若 $R(u, E_\alpha(i)) = (v, E_\beta(j))$, 出自顶点 $\langle u, i \rangle \in [N] \times [D]$ 的第 $\langle \alpha, \beta \rangle$ 条边依附于顶点 $\langle u, i \rangle$ 。直观上, 基于 $\langle \alpha, \beta \rangle$, 首先利用 G 中的步骤从 $\langle u, i \rangle$ 到 $\langle u, E_\alpha(i) \rangle$,

再将 $\langle u, E_\alpha(i) \rangle \equiv (u, E_\alpha(i))$ 视为 G' 的一个将其旋转为 $(v, j') \stackrel{\text{def}}{=} R'(u, E_\alpha(i))$ 的边 (即经过了 G' 中的步骤), 最后通过 G 中步骤从 $\langle v, j' \rangle$ 到 $\langle v, E_\beta(j') \rangle$ (定义 $j = E_\beta(j')$ 并使用 $j' = E_\beta(E_\beta(j')) = E_\beta(j)$)。

显然, 图 $G' \otimes G$ 是 d^2 -正则的且有 $D \cdot N$ 个顶点。参考文献[191]中证明 (使用了如附录 E.2.1.3 的技术) 的关键是, zig-zag 积的相对特征值的上界由两个图形的相对特征值的总和来界定 (即 $\bar{\lambda}(G' \otimes G) \leq \bar{\lambda}(G') + \bar{\lambda}(G)$, 其中 $\bar{\lambda}(\cdot)$ 表示的相关图的相对特征值)。如果 G' 和 G 为扩张图, 则 “ $G' \otimes G$ 为一个扩张图” 是非常直观的 (定性) 事实 (如考虑 G' 或 G 若是子图会有什么结果)。如果考虑记为 $G' \odot G$ 的 G' 或 G 的 (相关) 替代积 (Replacement Product), 则会更加直观, 其中当且仅当 $u = v$ 且 $(i, j) \in E$ 或依附于 u 的第 i 条边和依附于 v 的第 j 条边相等时, $\langle u, i \rangle \in [N] \times [D]$ 和 $\langle v, j \rangle$ 之间存在一个边。

迭代构造

迭代扩张图构造采用上述提到的 zig-zag 积以及图的平方。特别地, 构造开始于 d^2 -regular 图 $G_1 = G^2 = ([D], E^2)$, ^① 其中 $D = d^4$, $\bar{\lambda}(G) < 1/4$, 且按如下方式迭代: $G_{i+1} = G_i^2 \otimes G$, $i = 1, 2, \dots, t-1$ 。也就是说, 在每次迭代中, 首先求当前图的平方, 然后与给定的 (d -正则 D -顶点) 图 G 经由 zig-zag 积组合。这个过程保持以下两个变量:

(1) 图 G_i 是 d_2 -正则的, 且有 D^i 个顶点 (度数的界遵循如下的事实: d -正则图的 zig-zag 积总是产生一个 d_2 -正则图)。

(2) G_i 的相对特征值小于 $1/2$ (这里应用了 $\bar{\lambda}(G_{i-1}^2 \otimes G) \leq \bar{\lambda}(G_{i-1}^2) + \bar{\lambda}(G)$, 它又等于 $\bar{\lambda}(G_{i-1}^2) + \bar{\lambda}(G) < (1/2)^2 + 1/4$)。请注意, 通过与 G 的 zig-zag 积相加之前, 对 G_i 求平方来减少相对特征值)。

为确保可以构造 G_i , 应该证明, 实际上可以构造对应于它的边集的边旋转函数。这可归结为证明, 给定 G_{i-1} 的边的旋转函数, 可以计算 G_{i-1}^2 的边的旋转函数以及它与 G 的 zig-zag 积。注意到, 此计算相当于两个关于 G_{i-1} 的计算的递归调用 (和两个对应于恒定图 G 的计算)。但由于递归深度是最终图规模的对数级, 在递归运算中的时间开销是最终图规模的多项式。这对明确性的最小概念是足够了, 但不能达到最强的明确性。

强的明确性版本

为实现强的明确性构造, 稍微修改一下迭代结构。放弃 $G_{i+1}^2 = G_i^2 \otimes G$, 而采用 $G_{i+1}^2 = (G_i \times G_i)^2 \otimes G$, 其中 $G' \times G'$ 表示对 G' 与自身的张量积, 也就是说, 如果 $G' = (V', E')$, 则 $G' \times G' = (V' \times V', E'')$, 其中

$$E'' = \{(\langle u_1 u_2 \rangle, \langle v_1 v_2 \rangle) : (\langle u_1 v_1 \rangle, \langle u_2 v_2 \rangle) \in E'\}$$

相应的边旋转函数为

$$R'' = (\langle u_1 u_2 \rangle, \langle i_1 i_2 \rangle) = (\langle v_1, v_2 \rangle, \langle j_1, j_2 \rangle)$$

其中 $R'(u_1 i_1) = (v_1, j_1)$, $R'(u_2 i_2) = (v_2, j_2)$ (仍使用 $G_1 = G^2$)。基于该张量积保持了相对特征值 (同时平方度) 的事实, 同时利用一个含 $D = d^8$ 个顶点的 d -正则图 $G = ([D], E)$ 。^② 注意到由改进的

① 回顾对足够大的 d , 首先通过穷尽搜索找到 d -正则图 $G = ([d^4], E)$, 满足 $\bar{\lambda}(G) < 1/4$ 。

② 这种变化的原因是 $(G_i \times G_i)^2$ 将是 d^8 -正则的, 因为 G_i 是 d^2 -正则的。

迭代 $G_{i+1}=(G_i \times G_i) \otimes G$ 可得到含 $(D^{2^{i-1}})^2 \cdot D = D^{2^i-1}$ 个顶点的 d -正则图, 且 $\bar{\lambda}_2(G_i) < 1/2$ (因为 $\bar{\lambda}_2(G_{i-1} \times G_{i-1})^2 \otimes G \leq \bar{\lambda}_2(G_{i-1})^2 + \bar{\lambda}_2(G)$)。计算 G_i 顶点的邻居可以归结为计算一个有关 G_{i-1} 的常数, 但由于张量积运算的递归深度是最终图规模的双对数 (于是, 也是其中顶点的描述长度的对数)。

思考

在第一种构造中, zig-zag 积既用于增加图的规模, 也用于降低其度。然而, 由第二种构造 (其中, 图的张量积是增加图规模的主要载体) 所示, zig-zag 积的主要作用是降低图的度。图规模的增加只是一个副作用^①。在这两种情况下, 图的平方用于以补偿由 zig-zag 积所造成的相对特征值的轻微增加。回想第二构造是“正确”的, 因为它分离了 3 种不同的效果, 并且使用自然的操作来得到它们: 由图的张量积增加图的规模 (这反过来又增加了度), 由 zig-zag 积来降低度 (这反过来又增加了相对特征值), 又对图求平方用于减少相对特征值。

关于 zig-zag 积效应的更强界限

前面的描述基于在参考文献[191]中证明的事实, 即 zig-zag 积的相对特征值的上界为两个图的相对特征值 (即 $\bar{\lambda}(G' \otimes G) \leq \bar{\lambda}(G') + \bar{\lambda}(G)$)。其实, 参考文献[191]中证明了一个更强的上界, 即 $\bar{\lambda}(G' \otimes G) \leq f(\bar{\lambda}(G'), \bar{\lambda}(G))$, 其中

$$f(x, y) \stackrel{\text{def}}{=} \frac{(1-y^2) \cdot x}{2} + \sqrt{\left(\frac{(1-y^2) \cdot x}{2}\right)^2 + y^2} \quad (\text{E.10})$$

事实上, $f(x, y) \leq (1-y^2) \cdot x + y \leq x + y$ 。另一方面, 对于 $x \leq 1$, 有

$$f(x, y) \leq \frac{(1-y^2) \cdot x}{2} + \frac{1+y^2}{2} = 1 - \frac{(1-y^2) \cdot (1-x)}{2}$$

这蕴含了

$$\bar{\lambda}(G' \otimes G) \leq 1 - \frac{(1-\bar{\lambda}(G))^2 \cdot (1-\bar{\lambda}(G'))}{2} \quad (\text{E.11})$$

因此, $1-\bar{\lambda}(G' \otimes G) \geq (1-\bar{\lambda}(G))^2 \cdot (1-\bar{\lambda}(G'))/2$, 从而如果两个图都有正的特征值缝隙, 则其 Zig-Zag 积也有正的特征值缝隙 (即如果 $\lambda(G) < 1$ 和 $\lambda(G') < 1$ 均成立, 则 $\lambda(G' \otimes G) < 1$)。进一步地, 如果 $\bar{\lambda}(G) < 1/\sqrt{3}$, 则 $1-\bar{\lambda}(G' \otimes G) > (1-\bar{\lambda}(G'))/3$ 。这个事实在定理 5.6 的证明中起重要作用。

^① 我们指出, 实际上, 在某些应用中可能并不期望这种副作用。例如, 5.2.4 节中宁愿让图的规模增加, 也不能容忍常数规模的放大 (由常数规模图上的 zig-zag 积引起)。

附录 F

一些省略的证明

绅士的话比证明更可信，但你不是绅士，所以请给出证明。

莱昂纳德·莱维(1986)

由于种种原因（如过于专业化，或者超出了范围），本附录中的证明方法没有包括在正文中。另一方面，本书对证明的表述方法与原始方法和/或标准方法有较大差别，所以有必要给出这些证明。

内容提要

本附录包含对如下结论的证明：

(1) 通过随机化的 Karp-归约，可将 $\mathcal{PH}$ 归约为 $\#P$ （实际上是 $\oplus P$ ）。该证明遵循了 Toda 原始证明的思想，但实际表述方法有很大不同。

(2) 对任意满足 $f(n) \in \{2, 3, \dots, \text{poly}(n)\}$ 的整数函数 f ，有 $IP(f) \subseteq \mathcal{AM}(O(f))$ 和 $\mathcal{AM}(O(f)) \subseteq \mathcal{AM}(f)$ 。该证明只在一些次要的细节上不同于原始证明（分别在参考文献[21]和参考文献[107]中给出），但这些细节似乎很重要。

F.1 证明 $\mathcal{PH}$ 可归约为 $\#P$

回顾定理 6.16，其中称 $\mathcal{PH}$ 可以 Cook-归约为 $\#P$ 的（通过确定性归约）。这里要证明一个紧密相关的结论（也由 Toda^[212]提出），该结论放宽了对归约的要求（允许随机化），但对看似较弱的类使用预言机。后一个类被表示成 $\oplus P$ ，它是 $\#P$ 的“模 2 类比”。具体地，如果存在一个函数 $g \in \#P$ ，使得对任意 x 均有 $f(x) = g(x) \bmod 2$ ，则 f 属于 $\oplus P$ 类。同样，如果存在搜索问题 $R \in \mathcal{PC}$ ，使得 $f(x) = |R(x)| \bmod 2$ ，其中 $R(x) = \{y : (x, y) \in R\}$ ，则 f 属于 $\oplus P$ 类（ $\oplus P$ 中的符号 $\oplus$ 实际上表示奇偶性，也就是模 2 加运算。事实上，用一个形如 $\#_2 P$ 的记号会更恰当）。

定理 F.1 通过一个概率多项式时间归约， $\mathcal{PH}$ 中任意集合均可归约到 $\oplus P$ ，且归约为多到一映射，其错误概率可以忽略。

该证明遵循参考文献[220]中原始证明的根本思想，但实际表述完全不同。参考文献[136, 212]中给出了另一种证明。

教学注释：证明定理 F.1 的非一致版本是很容易的，其中断言 AC0 电路可以用包含无限个合取对的电路来估算，其中每个合取具有多项式-对数的扇入。用这种方法证明定理 F.1 需要仔细考虑实现问题，并利用习题 3.8 中的转换类型。另外，这种表述容易使其中的概念性步骤含义不清。

证明概要：证明包含三部分。第一部分利用了 $\mathcal{NP}$ 可利用概率的 Karp-归约，归约到 $\oplus P$ 的事实，且该归约是相对的（即对任意预言 A ，将 $\mathcal{NP}^A$ 归约为 $\oplus P^A$ ）。^①第二部分，这里允许错误缩减（即在到 $\oplus P$ 的随机归约中），从而使归约的错误概率呈指数降低。^②第三部分将第一部分扩展至 Σ_k ，其中利用了命题 3.9，以及上述的错误缩减。这些部分对应于以下的 3 个证明步骤。

第一步：构造从 $\mathcal{NP}$ 到 $\oplus P$ 的随机 Karp-归约。

第二步：减少上述 Karp-归约的错误概率，使错误概率呈指数降低。这种低错误概率对下一步起关键作用。

第三步：扩展第一步的归约，证明 Σ_2 可随机归约为 $\oplus P$ （同时利用第二步）。直观上，对任意预言 A ，第一步中的归约提供了 $\mathcal{NP}^A$ 到 $\oplus P^A$ 的归约，而 A 到 B 的归约具有指数降低的错误概率，这样可以得到 $\oplus P^A$ 到 $\oplus P^B$ 的归约（或类似地，将 $\mathcal{NP}^A$ 归约到 $\mathcal{NP}^B$ ）。注意到， $\oplus P^{\oplus P} = \oplus P$ ，从而得到 Σ_2 （视为 $\mathcal{NP}^{\mathcal{NP}}$ ）到 $\oplus P$ 的随机 Karp-归约。

在完成第三步时，应该提供一般情况下（对任意 $k \geq 2$ ，将 Σ_k 随机归约为 $\oplus P$ ）的所有步骤。在完成证明时，需要将 Σ_2 的情形扩展到一般的 Σ_k （对任意 $k \geq 2$ ）。这个扩展实际上很复杂，但其思想已体现在 Σ_2 情形中。另外，我们认为 Σ_2 情形本身也是很有意义的。

教学注释：上述第三步建议将 $\mathcal{NP}^A$ 及 P^B 等概念抽象化。我们更倾向于将第三步表述成是第一步的扩展（利用第二步）。这是明确实现第一步（即提出 $\mathcal{NP}$ 到 P 的随机 Karp-归约）的一个原因。注意：利用判定 PC 中某些关系 R 的唯一解的困难性，直接可得到第一步（ $\mathcal{NP}$ 到 P 的归约），这是因为利用恒等映射，承诺问题 $(US_R, \bar{S}_R)$ 可归约为 $\oplus R = \{x : |R(x)| \equiv 1 \pmod{2}\}$ ，其中 $US_R = \{x : |R(x)| = 1\}$ ， $\bar{S}_R = \{x : |R(x)| = 0\}$ 。然而，为了自包含及概念上的准确性，我们提出另一种证明。

第一步：直接证明 $\mathcal{NP}$ 情形

与定理 6.29 的证明相同，从任意 $R \in PC$ 开始，目标是通过随机 Karp-归约^③将 $S_R = \{x : |R(x)| \geq 1\}$ 归约到 $\oplus P$ 。构造这种归约的标准方法（如参考文献[136,178,212,220]）只需要利用定理 6.29 证明中使用的归约，但我们认为这种方法在概念上是错误的，解释如下。

回顾定理 6.29 的证明中包括了执行一个具有如下性质的随机筛，对任意 $x \in S_R$ ， $R(x)$ 中单个元素能以显著概率通过筛（对于唯一解问题，可由一个预言机检测该事件）。的确， P 中

① 事实上，“相对性”要求假设 $\mathcal{NP}$ 和 $\oplus P$ 均与一类（标准的）机器相关，该机器可扩展为相应的预言机（见 3.2.2 节的说明）。利用判定相应 PC 中关系的（确定的多项式时间）机器，可以证明这个假定对于两个类都成立。另外，还可以利用一个事实，即上述归约在某种多项式时间计算的谓词 ψ 的意义下是“高结构性”的，归约将 x 映射为 $\langle x, s \rangle$ ，使得对任意非空集 $S_x \subseteq \{0,1\}^{p(|x|)}$ ，

有 $\Pr_x \left[\left| \{y \in S_x : \psi(x, s, y)\} \right| \equiv 1 \pmod{2} \right] > 1/3$ 。

② 这样的错误缩减在到唯一解问题的归约中是不允许的。考虑到 $\mathcal{NP}$ 到 $\oplus P$ 的归约与 $\mathcal{NP}$ 到唯一解问题归约间的相似性。

③ 如定理 6.29，如果 PC 中任意搜索问题可简化地归约为 R ，则可将 S_R 归约为 $\oplus R$ 。具体地，我们将证明对某个 $R_2 \in PC$ ， S_{R_2} 可随机归约为 $\oplus R_2$ ，由此得到 S_R 到 $\oplus R$ 的归约（利用 R_2 到 R 的简化归约）。

的充分预言可以正确检测到 $R(x)$ 中单个元素是否通过筛。然而，由定义可知，该预言机能正确地检测一种更一般的情形，即 $R(x)$ 中奇数个元素通过筛。因此，坚持使用只允许 $R(x)$ 中单个元素通过的随机筛，似乎有点过度（或者至少是概念上有误）。取而代之的，应该使用一个不太严格的随机筛，它能以显著的概率允许 $R(x)$ 中奇数个元素通过。实现这个筛的工具是小偏移发生器（见 8.5.2 节）。

实际上，是要利用一个小偏移发生器来筛选可能的解，从而将 S_R 随机归约为 $\mathcal{P}$ 。直观上，将 x 随机地归约为 $\langle x, s \rangle$ ，其中 s 为发生器的随机种子， y 是实例 $\langle x, s \rangle$ 的解当且仅当 $y \in R(x)$ 且 $G(s)$ 的第 y 比特为 1（事实上，如果 $|R(x)| \geq 1$ ，则实例 $\langle x, s \rangle$ 有奇数个解的概率约为 $1/2$ ，而当 $|R(x)| = 0$ 时， $\langle x, s \rangle$ 无解）。具体地，我们使用一个强的高效发生器（见 8.5.2.1 节），记作 $G: \{0,1\}^k \rightarrow \{0,1\}^{\ell(k)}$ ，其中 $G(U_k)$ 最多有 $1/6$ 的偏差且 $\ell(k) = \exp(\Omega(k))$ 。也就是说，给定种子 $s \in \{0,1\}^k$ 和索引 $i \in [\ell(k)]$ ，能在多项式时间内生成 $G(s)$ 的第 i 比特，记作 $G(s, i)$ 。不失一般性，假设对某些多项式 p ， $R(x) \subseteq \{0,1\}^{P(|x|)}$ ，考虑关系

$$R_2 \stackrel{\text{def}}{=} \{(\langle x, s \rangle, y) : (x, y) \in R \wedge G(s, y) = 1\} \quad (\text{F.1})$$

其中 $y \in \{0,1\}^{P(|x|)} \equiv [2^{P(|x|)}]$ ， $s \in \{0,1\}^{O(|y|)}$ 使得 $\ell(|s|) = 2^{|y|}$ 。换言之， $R_2(\langle x, s \rangle) = \{y : y \in R(x) \wedge G(s, y) = 1\}$ 。然后，对任意 $x \in S_R$ ，均匀选择的 $s \in \{0,1\}^{O(|y|)}$ 以至少 $1/3$ 的概率满足 $|R_2(\langle x, s \rangle)| \equiv 1 \pmod{2}$ ，然而，对任意 $x \notin S_R$ 和任意 $s \in \{0,1\}^{O(|y|)}$ ，有 $|R_2(\langle x, s \rangle)| = 0$ 。这里的关键是 $R_2 \in \text{PC}$ （从而 $\oplus R_2$ 属于 $\oplus \mathcal{P}$ ）。因此， S_R 的成员判定可随机归约为 $\oplus R_2$ （通过 x 到 $\langle x, s \rangle$ 的多到一随机映射，其中 s 在 $\{0,1\}^{O(P(|x|))}$ 中均匀选择）。由于上述结论对任意 $R \in \text{PC}$ 成立，从而通过随机的 Karp-归约，可将 $\mathcal{NP}$ 归约为 $\oplus \mathcal{P}$ 。

coNP 情形

可以利用 Cook-归约将 coNP 归约到 $\mathcal{NP}$ ，从而证明 coNP 可随机归约到 $\oplus \mathcal{P}$ ，但是要强化一个事实，即随机 Karp-归约也能做到这一点。从 $\mathcal{NP}$ 中集合的归约出发，注意到，对于 $S \in \text{coNP}$ （即 $S = \{x : R(x) = \emptyset\}$ ），可以得到关系 R_2 ，使得 $|R_2(\langle x, \cdot \rangle)| \equiv 0 \pmod{2}$ 蕴含了 $x \in S$ 。我们希望反过来，令 $|R_2(\langle x, \cdot \rangle)| \equiv 1 \pmod{2}$ 蕴含 $x \in S$ ，为此，扩张关系 R_2 ，为每个 x 增加一个“哑”解。例如，可将 $R_2(\langle x, s \rangle)$ 重新定义为 $\{0y : y \in R_2(\langle x, s \rangle)\} \cup \{10^{P(|x|)}\}$ 。事实上，刚才已经证明并利用了一个事实，即 $\oplus \mathcal{P}$ 在求补运算是封闭的。

注意到，只有当归约为 $\oplus \mathcal{P}$ 而非 $\# \mathcal{P}$ 时，分别处理 $\mathcal{NP}$ 和 coNP 的情形才有价值。相反，即使是 Σ_2 到 $\# \mathcal{P}$ 的归约也是有意义的，从而 Σ_2 到 $\oplus \mathcal{P}$ （见第三步）的归约有价值。该归约很大程度上依赖于到 $\oplus \mathcal{P}$ 的随机 Karp-归约中可以使用错误缩减。

第二步 错误缩减

针对证明核心的一个重要观察是，在到 $\oplus \mathcal{P}$ 的随机 Karp-归约中可以大大减少（单边）错误概率。具体地，令 R_2 如式 (F.1)， t 为任意多项式，则满足下式的二元关系 $R_2^{(t)}$ 可以提供错误缩减：

$$\left| R_2^{(t)} \left(\left\langle x, s_1, \dots, s_{t(|x|)} \right\rangle \right) \right| = 1 + \prod_{i=1}^{t(|x|)} (1 + |R_2(\langle x, s_i \rangle)|) \quad (\text{F.2})$$

这是由于 $\left| R_2^{(t)} \left(\langle x, s_1, \dots, s_{t(|x|)} \rangle \right) \right|$ 为奇数当且仅当对某个 $i \in [t(|x|)]$, $|R_2(\langle x, s_i \rangle)|$ 为奇数。因此, 有

$$P_{s_1, s_2, \dots, s_{t(|x|)}} \left[\left| R_2^{(t)} \left(\langle x, s_1, s_2, \dots, s_{t(|x|)} \rangle \right) \right| \equiv 0 \pmod{2} \right] = P_s \left[\left| R_2(\langle x, s \rangle) \right| \equiv 0 \pmod{2} \right]^{t(|x|)}$$

其中 $s, s_1, s_2, \dots, s_{t(|x|)}$ 在 $\{0, 1\}^{O(p(|x|))}$ 上均匀、独立分布 (且 p 满足 $R(x) \subseteq \{0, 1\}^{p(|x|)}$)。这意味着, 在 S_R 到 $\oplus R_2$ 的随机归约 (将 x 映射为 $\langle x, s \rangle$) 中, 单边错误概率可以大大减小, 方法是将 S_R 归约为 $\oplus R_2^{(t)}$, 归约将 x 映射到 $\langle x, s_1, s_2, \dots, s_{t(|x|)} \rangle$ 。具体而言, 当我们希望得到一个“奇结果” (即 $x \in S_R$) 时, 错误概率 ε (如 $\varepsilon=2/3$) 被降低至 ε^t , 而当希望得到“偶结果” (即 $x \in \bar{S}_R$) 时, 仍保持错误概率为 0。

关键在于 $\oplus R_2^{(t)}$ 是否属于 $\oplus \mathcal{P}$, 也就是说, $R_2^{(t)}$ (如式 (F.2) 中假设的) 是否可在 $\mathcal{PC}$ 中实现。回答是肯定的, 这可利用笛卡儿积构造 (并加入某些哑解) 来证明。例如, 令 $R_2^{(t)} \left(\langle x, s_1, s_2, \dots, s_{t(|x|)} \rangle \right)$ 中包含一些 $\langle \sigma_0, y_1, y_2, \dots, y_{t(|x|)} \rangle$, 使得或者 $\sigma_0=1$ 且 $y_1=\dots=y_{t(|x|)}=0^{p(|x|)+1}$, 或者 $\sigma_0=0$ 且对任意 $i \in [t(|x|)]$, 有 $y_i \in (\{0\} \times R_2(\langle x, s_i \rangle)) \cup \{10^{p(|x|)}\}$ (即或者 $y_i=10^{p(|x|)}$, 或者 $y_i=0y'_i$ 且 $y'_i \in R_2(\langle x, s_i \rangle)$)。

我们要强调, 在从如式 (F.1) 的 R_2 出发时, 通过上述错误缩减过程可以得到一个上限为 $\exp(-q(|x|))$ 的错误概率, 其中 q 为任意多项式。很快将看到这个说明的重要性。

第三步: Σ_2 情形

通过上面的预备过程, 我们已经准备好处理 $s \in \Sigma_2$ 的情形。由命题 3.9 可知, 存在一个多项式 p 和集合 $S' \in \Pi_1 = \text{co}\mathcal{NP}$ 使得 $S = \{x: \exists y \in \{0, 1\}^{p(|x|)} \text{ s.t. } (x, y) \in S'\}$ 。根据 $S' \in \text{co}\mathcal{NP}$, 并应用上述 S' 到 $\oplus \mathcal{P}$ 的归约以及充分的错误缩减, 可得到错误概率上限为 $\varepsilon \cdot 2^{-p(|x|)}$, 其中 $\varepsilon \leq 1/7$ 未定义 (对于 Σ_2 情形, 令 $\varepsilon=1/7$ 即可, 但是对于 Σ_k , 需要更小的值)。因此, 对于充分的多项式 t (即 $t(n+p(n))=O(p(n)\log(1/\varepsilon))$), 可以得到关系 $R_2^{(t)} \in \mathcal{PC}$, 满足对任意 x 及 $y \in \{0, 1\}^{\text{poly}(|x|)}$, 对于随机选择的 $s' \in \{0, 1\}^{\text{poly}(|x|)}$, 当且仅当 $|R_2^{(t)}(\langle x', s' \rangle)|$ 为奇数时, $x' \stackrel{\text{def}}{=} (x, y) \in S'$, 这以至少 $1-\varepsilon \cdot 2^{-p(|x|)}$ 的概率成立。^①

利用联合概率 (对所有可能的 $y \in \{0, 1\}^{p(|x|)}$) 可以得到, 对所有选择的 s' , 以至少 $1-\varepsilon$ 的概率有, $x \in S$ 当且仅当存在一个 y 使得 $|R_2^{(t)}(\langle (x, y), s' \rangle)|$ 为奇数。现在, 与对 $\mathcal{NP}$ 的处理相同, 我们希望将后者归约为 $\oplus \mathcal{P}$ 中的“存在问题”。也就是说, 希望定义一个关系 $R_3 \in \mathcal{PC}$,

① 回顾 $|s'|=t(|x|) \cdot O(p(|x|))$, 其中 $R'(x') \subseteq \{0, 1\}^{p(|x|)}$ 是 S' 的“证据关系” (即 $x' \in S'$ 当且仅当 $R'(x') \subseteq \{0, 1\}^{p(|x|)}$)。因此, $R_2(\langle x', s' \rangle) \subseteq \{0, 1\}^{p(|x|)+1}$ 和 $R_2^{(t)}(\langle x', s' \rangle)$ 是 $\{0, 1\}^{1+t(|x|)(p(|x|)+2)}$ 的子集。注意 (由于我们是从 $S' \in \text{co}\mathcal{NP}$ 出发的): 错误只发生在 S' 的“否”实例中, 而“是”实例总是被接受。然而, 为简化阐述, 我们允许 S' 的“是”实例中也有错误。这并不重要, 因为总之在 S 的“是”实例中也是有错误的。

使得对于随机选择的 s , $|R_3(\langle x, s, s' \rangle)| \bmod 2$ 的值暗示了是否 $x \in S$ (通过暗示是否存在一个 y 使得 $|R_2^{(t)}(\langle (x, y), s' \rangle)|$ 为奇数)。类似于式 (F.1), 考虑二元关系

$$I_3 \stackrel{\text{def}}{=} \left\{ \langle \langle x, s, s' \rangle, y \rangle : |R_2^{(t)}(\langle (x, y), s' \rangle)| \equiv 1 \pmod{2} \wedge G(s, y) = 1 \right\} \quad (\text{F.3})$$

换言之, $I_3(\langle x, s, s' \rangle) = \{y : |R_2^{(t)}(\langle (x, y), s' \rangle)| \equiv 1 \pmod{2} \wedge G(s, y) = 1\}$ 。的确, 如果 $x \in S$, 那么, 在随机选择的 s' 中, 以至少 $1 - \varepsilon$ 的概率, 以及在随机选择的 s 中, 以至少 $1/3$ 的概率有 $|I_3(\langle x, s, s' \rangle)|$ 为奇, 而对任意 $x \notin S$ 及所有 s , 有 $P_{r_s} \left[|I_3(\langle x, s, s' \rangle)| = 0 \right] \geq 1 - \varepsilon$ ①。注意: 当 $\varepsilon \leq 1/7$ 时, 对任意 $x \in S$, 有 $P_{r_{s,s}} \left[|I_3(\langle x, s, s' \rangle)| \equiv 1 \pmod{2} \right] \geq (1 - \varepsilon)/3 \geq 2/7$, 而对任意 $x \notin S$, 有 $P_{r_{s,s}} \left[|I_3(\langle x, s, s' \rangle)| \equiv 1 \pmod{2} \right] \leq \varepsilon \leq 1/7$ 。因此, $|I_3(\langle x, \cdot, \cdot \rangle)| \bmod 2$ 随机地指示着是否 $x \in S$, 但是尚不清楚是否 I_3 属于 PC (事实上, I_3 很可能不属于 PC)。关键在于, 存在 $R_3 \in PC$ 使得 $\oplus R_3 = \oplus I_3$ 。

具体地, 考虑

$$R_3 \stackrel{\text{def}}{=} \left\{ \langle \langle x, s, s' \rangle, \langle y, z \rangle \rangle : \langle \langle (x, y), s' \rangle, z \rangle \in R_2^{(t)} \wedge G(s, y) = 1 \right\} \quad (\text{F.4})$$

其中 $\langle y, z \rangle \in \{0, 1\}^{p(|x|)} \times \{0, 1\}^{\text{poly}(|x|)}$ (也就是说, 当 $\langle \langle (x, y), s' \rangle, z \rangle \in R_2^{(t)}$ 且 $G(s, y) = 1$ 时, $\langle y, z \rangle$ 属于 $R_3(\langle x, s, s' \rangle)$)。显然, $R_3 \in PC$, 从而剩下的就是要证明 $|R_3(\langle x, s, s' \rangle)| \equiv |I_3(\langle x, s, s' \rangle)| \pmod{2}$ 。令 $\chi_{y,z}$ (或 ξ_y) 表示事件 $\langle \langle (x, y), s' \rangle, z \rangle \in R_2^{(t)}$ (或事件 $G(s, y) = 1$) , 并注意到

$$\begin{aligned} |R_3(\langle x, s, s' \rangle)| \bmod 2 &= \oplus_{y,z} (\chi_{y,z} \wedge \xi_y) \\ |I_3(\langle x, s, s' \rangle)| \bmod 2 &= \oplus_y ((\oplus_z \chi_{y,z}) \wedge \xi_y) \end{aligned}$$

利用两个对应布尔表达式的等价性, 断言得证。从而, S 可以随机地 Karp-归约为 $\oplus R_3 \in \oplus P$ (由 x 到 $\langle x, s, s' \rangle$ 的多到一随机映射, 其中 (s, s') 在 $\{0, 1\}^{O(p(|x|))} \times \{0, 1\}^{\text{poly}(|x|)}$ 中均匀选择)。由于这对任意 $S \in \Sigma_2$ 都成立, 于是得到结论, Σ_2 可以随机地 Karp-归约为 $\oplus P$ 。

另外, 该归约 (Σ_2 到 $\oplus P$) 中可以使用错误缩减, 从而使得到的归约能应用于 Σ_3 (被视为 $\mathcal{NP}^{\Sigma_2}$) 的情形。技术上的难点是上述归约具有双边错误, 其中一种类型 (或“边”) 的错误来自于 $S' \in \text{coNP}$ 到 $\oplus R_2^{(t)}$ 的归约 (对 S' 的“否”实例出错), 另一种类型 (或“边”) 的错误来自于 (新的) S 到 $\oplus R_3$ 的归约 (对 S 的“是”实例出错)。然而, 第一个归约中的错误概率 (可以令其) 非常小, 当对第二个归约使用错误缩减时, 可被忽略, 见以下说明。

① 我们注意到, 实际上, 如果 $x \in S$, 则存在 y , 使得且对任意选择的 s' , 有 $|R_2^{(t)}(\langle (x, y), s' \rangle)|$ 为奇 (因为从 $S' \in \text{coNP}$ 到 $\oplus P$ 的归约对于“是”实例错误为 0)。另一方面, 如果 $x \notin S$, 则 $P_{r_s} \left[(\forall y) |R_2^{(t)}(\langle (x, y), s' \rangle)| \equiv 0 \pmod{2} \right] \geq 1 - \varepsilon$ (因为对任意 y , 有 $(x, y) \notin S'$, 且从 coNP 到 $\oplus P$ 的归约对于“否”实例错误不为 0)。因此, 对任意的 $x \notin S$ 和 s , 有 $P_{r_s} \left[|I_3(\langle x, s, s' \rangle)| = 0 \right] \geq 1 - \varepsilon$ (因为从 $S \in \mathcal{NP}^S$ 到 $\oplus P^S$ 的归约对于“否”实例错误概率为 0)。总结起来, 联合归约具有双边错误, 因为两个归约从不同方向引入了错误。

一般情形

首先要注意, 与 coNP 情况相同, 我们可以得到 $\Pi_2 = \text{co}\Sigma_2$ 中集合的类似归约 (到 $\oplus P$)。事实上, 要证明如何利用 Σ_{k-1} (或 Π_{k-1}) 的归约将 Σ_k 归约至 $\oplus P$ 。具体地, 在处理 $S \in \Sigma_k$ 时, 考虑多项式 p 和集合 $S' \in \Pi_{k-1}$, 满足 $S = \{x : \exists y \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,y) \in S'\}$ 。根据对 Π_{k-1} 的处理结果, 利用关系 $R_k^{(t_k)}$, 满足关于 s' 的选择, $|R_k^{(t_k)}(\langle (x,y), s' \rangle)| \bmod 2$ 的值以显著的高概率蕴含着是否 $(x,y) \in S'$ 。利用对 NP (及 Σ_2) 的处理思想, 检查是否存在 $y \in \{0,1\}^{p(|x|)}$, 使得 $|R_k^{(t_k)}(\langle (x,y), s' \rangle)| \equiv 1 \bmod 2$ 。这就得到一个关系 R_{k+1} , 满足对随机的 s, s' , $|R_{k+1}(\langle x, s, s' \rangle)| \bmod 2$ 的值蕴含着是否 $x \in S$ 。最后, 利用错误缩减, 并忽略 s' 为“坏”的概率, 得到期望的关系 $R_{k+1}^{(t_{k+1})}$ 。

上述的归纳步骤在实现时需要考虑一些问题。具体地, 如果希望求 (S) 到 $\oplus R_{k+1}^{(t_{k+1})}$ 归约的错误概率 ε_{k+1} 上限, 则 (S') 到 $R_k^{(t_k)}$ 的归约错误概率上限 ε_k 应该满足 $\varepsilon_k \leq \varepsilon_{k+1} \cdot 2^{-p(|x|)}$ (且对 t_k 做相应设置)。因此, 证明 PH 可随机归约为 $\oplus P$ 实际上 (至少部分) 是“自顶向下”的; 也就是说, 从任意的 $S \in \Sigma_k$ 开始, 首先确定辅助集合 (根据命题 3.9), 并证明这些集合 (位于 PH 的较低层) 间归约的错误上限, 只有做到了这些, 才能证明存在这样的归约。事实上, 后一个 (主要的) 步骤是“自底向上”的, 为了将第 $i+1$ 层的集合归约到 $\oplus P$, 利用了第 i 层集合 (到 $\oplus P$) 的归约。

F.2 证明 $IP(f) \subseteq \mathcal{AM}(O(f)) \subseteq \mathcal{AM}(f)$

使用 9.1.4.3 节的符号, 在这里重述两个结果。

定理 F.2 (利用 $\mathcal{AM}$ 圈-高效地模仿 IP) 设 $f: \mathbb{N} \rightarrow \mathbb{N}$ 为一个多项式界函数, 则 $IP(f) \subseteq \mathcal{AM}(f+3)$ 。

要说明的是, 根据以下对 $\mathcal{AM}$ 线性加速的圈复杂度, 足以证明 $IP(f) \subseteq \mathcal{AM}(O(f))$

定理 F.3 ($\mathcal{AM}$ 的线性加速) 设 $f: \mathbb{N} \rightarrow \mathbb{N}$ 为一个多项式界函数, 则 $\mathcal{AM}(2f) \subseteq \mathcal{AM}(f+1)$ 。

结合以上两个定理, 可得到一个对 IP 的线性加速; 也就是说, 对任意多项式界限的 $f: \mathbb{N} \rightarrow (\mathbb{N} \setminus \{1\})$, 有 $IP(O(f)) \subseteq \mathcal{AM}(f) \subseteq IP(f)$ 。本附录要证明上述两个定理。

注意: 定理 F.2 的证明依赖于一个事实, 即对任意 f , $IP(f)$ 的错误缩减是可能的。具体地, 利用并行重复 (见参考文献[90]中的附录 C.1) 可实现错误缩减。我们指出, 定理 F.3 的证明中也隐含了 (关于 $\mathcal{AM}(f)$ 的) 错误缩减 (参考文献[23]的原始证明中则明确指出)。

F.2.1 利用 $\mathcal{AM}$ -游戏模拟通用的交互式证明

本节证明定理 F.2。与 Goldwasser 和 Sipser^[11]的原始证明相比, 我们的证明仅在概念和迭代模拟程序的实现上有所不同。

F.2.1.1 概述

我们的目标是将一个通用的交互式证明系统 (P, V) 转化为针对同一集合的公开掷硬币交

交互式证明系统。不失一般性, 假设 P 是关于 V 的一个优化证明者 (即 P 使 V 对任意输入的接受概率最大化), 则对任意的“是”实例 x , 使用随机数集合 A_x 的 V 在与证明者交互后, 接受该实例。对于 (与“是”实例等长的)“否”实例 x , 集合 A_x 要小得多。这里的基本思想是构造一个公开掷硬币系统, 其中对于公共输入 x , 证明者向验证者证明集合足够大。这样一个证明系统可利用近似计数的思想 (见定理 6.27 的证明) 来构造, 其中将 NP-预言用证明者代替, 要求证明者证明其应答的正确性。为实现这一思想, 要求深入研究使 V 接受一个输入的随机数集合。

一个极受限版本

在阐述上述方法的实现时, 首先考虑含两个消息的交互式证明系统的受限版本。回顾在一个二消息交互式证明系统中, 验证者 V 向证明者 P 发送一个消息 (基于公共输入和 V 的内部随机数), 证明者返回一个消息, V 判断是否接受该输入。我们进一步地将注意力集中于假设 V 的每个可能消息都是等概的, 且可能的消息数量易由输入确定。对于输入 x , 验证者 V 生成 $l = l(|x|)$ 个随机数, 从 $N = N(x)$ 个可能的消息中选择一个发送。注意: 如果 x 为“是”实例, 则对 V 的每个可能消息, 存在一个 P 应答, 可由 V 的 $2^l/N$ 个相应的随机数序列 (即令 V 发出该消息的随机数序列) 接受。另一方面, 如果 x 为“否”实例, 则对于均匀选择的 V 的消息, 优化的 P -应答只能被极少数随机数序列接受。我们要证明, 如何用公开掷硬币系统来模仿这样一个交互式证明系统。

在公开掷硬币系统中, 对于输入 x , 证明者试图对 (原始系统中) 每个可能的 V -消息, 证明存在 (原始证明者的) 一种应答, 可以被 V 的 $2^l/N$ 个相应的随机数序列接受。注意: 双方都易确定 $N = N(x)$ 和 $l = l(|x|)$, 所以如果上述断言成立, 则 x 必为“是”实例。新的交互运行如下: 首先, 验证者均匀选择 V 的一个随机数序列, 记作 r , 将其发送给证明者。序列 r 确定一个 V -消息, 记作 α 。然后, 证明者返回一个充分的 P -消息, 记作 β , 并向验证者交互式地证明 β 应该被 V 的对应于消息 α 的 $2^l/N$ 个随机数序列接受 (即 β 不仅要被 r 接受, 而且还要被对应于消息 α 的 $2^l/N$ 个随机数序列接受)。后一个交互式证明利用了定理 6.27 的证明思想: 验证者利用一个随机筛, 只令 $(2^l/N)^{-1}$ 的元素通过, 证明者证明存在足够多的 V 的随机数序列能通过筛 (只需提出这样一个序列)。^①我们强调上述交互过程 (特别是随机筛) 可在公开掷硬币模型下实现。

去除一个限制

下面我们去除一个限制, 即 V 的可能消息数量由输入易确定, 但仍假设所有消息是等概的。此时, 证明者要提供消息数量 N , 并要证明至少存在 N 个可能的 V -消息 (且与上述情况相同, 对每个 V 的消息, 存在一个 P -应答, 可被 V 的 $(2^l/N)^{-1}$ 个随机数序列接受)。也就是说, 证明者将对至少 N 个可能的 V -消息证明存在着被 V 的 $(2^l/N)^{-1}$ 个随机数序列接受的 P -应答。这要求两次调用上述的“下限”协议。也就是说, 第一方对所有可能的 V -消息应用一个随机筛, 使得这些消息中只有 N^{-1} 的部分通过, 然后双方对符合通过消息的随机数序列应用随机筛, 使得只有 $(2^l/N)^{-1}$ 的序列通过。

① 实际上, 验证者易检查随机数序列 r' 是否通过筛, 并检查其是否符合原始消息 α , 从而, 当证明者以 β 应答时, 令 V 接受 (即 V 在收到证明者消息 β 时, 对于随机数 r' , 应该接受输入)。

IP(2)的一般情形

处理一般形式的二消息交互式证明时,要求也去掉另一个限制,即所有可能的 V -消息等概出现。此时,证明者可以将所有的 V -消息聚集成几个 (l 个) 簇,使得每个簇中的消息以近似相同(最多相差 2-因子)的概率发送。主要考虑具有最大概率的簇,证明者的运行过程同上(即发送 i , 并称存在 $2^l/l$ 个可能的 V -消息,每个被 2^l 个随机数序列支持)。这需要分割“是”实例与“否”实例的概率缝隙,使其相差一个因子,该因子与簇个数及簇间近似程度(即是 $O(l)$ 的一个因子)的乘积有关,^①与原始缝隙(可由错误缩减得到)相比,这个缝隙是可以忽略的。

对 IP 所有层次的处理

目前,我们只处理了含两个消息的证明系统(即 $IP(2)$)。对一般情形, $IP(f)$ 的处理应该通过递归(或迭代)来实现,递归的每一层(或每次迭代)类似于 $IP(2)$ 的情形。回顾对于 $IP(2)$ 情形的处理,可归结为选择一个随机的 V -消息 α , 令证明者返回 P -应答 β , 并证明 β 可被 V 的许多随机数序列接受。换言之,证明者要证明在由 V -消息 α 所定义的条件概率空间中,原始验证者 V 以高概率接受。在一般情形(即 $IP(f)$) 中,后一个断言涉及到剩余交互中的接受概率,剩余交互中含 $f-2$ 个消息,从而可迭代使用同一个协议(直至最后一个消息,这就与 $IP(2)$ 相同)。唯一的问题在于,在剩余交互中,验证者随机选择一个消息可能并不容易(像严格受限版本中那样)。然而,在去除第一个限制时已经这样做了,验证者可以接受证明者的帮助,并保证自己没有被骗。这个过程在附录 F.2.1.2 中详细介绍了,其中充分定义了“随机选择”协议的概念(需要在公开掷硬币模型下实现)。为简单起见,我们考虑在(剩余)概率空间中均匀选择的随机数,因为这样的序列确定了所需要的随机 V -消息。

F.2.1.2 随机选择

文献中出现了各种类型的“随机选择”协议(如参考文献[227]中的 6.4 节)。这些协议的主题是允许概率多项式时间参与方(称为验证者)从集合 $S \subseteq \{0,1\}^\ell$ 中采样,采样时需要有另一个参与方(称为证明者)帮助,证明者有强大的计算能力,但不一定诚实。这些称呼符合交互式证明的习惯,且可在这些协议的典型实现中进一步合理化,实现时,这些协议被当作交互式证明系统(其中第一方作为上层程序中的验证者、第二方为上层程序中的证明者)的子程序。各种类型的随机选择协议差别在于协议的要求,以及对集合 S 的了解程度。

这里我们假设验证者得到了参数 N , 并设其等于 $|S|$, 协议的实现保证了仅当集合规模不超过 N 时才有意义。我们要求一个常数轮的(最好含两个消息)公开掷硬币协议(在这种环境下),关于安全参数 $\varepsilon \geq 1/\text{poly}(\ell)$, 满足以下两个条件:

(1) 如果双方都是诚实的且 $N = |S|$, 则验证者的输出与均匀分布在 S 上是 ε -接近的。另外,验证者总是输出 S 中的元素。

(2) 对任意 $S' \subseteq \{0,1\}^\ell$, 如果验证者遵循协议,则无论证明者行为如何,验证者的输出属于 S' 的概率至多为 $\text{poly}(\ell/\varepsilon) \cdot (|S'|/N)$ 。

实际上,第二个性质仅当集合 S' 的规模(远远)小于 N 时才有意义。在使用这样的协议时,设置 ε 为常量(如 $\varepsilon = 1/2$)。

^① 这个损失是由于概率的分布对所有实例并不相同,例如,在一种情况下(如对某些“是”实例),所有的簇可能有相同概率,从而去除相应的因子,而在另一些情况下(如对某些“否”实例),所有概率在一个簇中被加起来了。

满足上述条件的、含 3 次消息交互的公开掷硬币协议可利用构造 6.2 的思想来实现。具体地, 令 $m = \max(0, \log_2 N - O(\log \ell / \varepsilon))$, 以保证如果 $|S| = N$, 则由均匀选择的 Hash 函数所定义的 2^m 个单元包含 S 中 $(1 \pm \varepsilon) \cdot |S| / 2^m$ 个元素的概率极高。在协议中, 证明者任意选择一个好的 Hash 函数 (即可确定 S 的这种划分), 发送给验证者, 验证者返回一个均匀选择的单元, 证明者针对该单元的应答是 S 中均匀选择的一个元素, 且位于该单元中。^①

我们强调, 上述协议必须在公开掷硬币模型下, 并要注意其中使用了 3 个消息而不是两个, 当然, 这对实际应用的影响很小 (见附录 F.2.1.3)。事实上, 这个协议满足上述两条性质。特别地, 第二方能遵循协议是因为对任意可能的 Hash 函数, 包含 S' 中一个元素的单元所占比例至多为 $|S'| / 2^m$, 其上限是 $\text{poly}(\ell / \varepsilon) \cdot |S'| / N$ 。

F.2.1.3 迭代分割协议

利用附录 F.2.1.2 中的随机选择协议, 可以构造任意交互式证明系统 (P, V) 的公开掷硬币版本。首先给出符号说明。

固定 (P, V) 的任意输入 x , 将交互中的消息对数记作 $t = t(|x|)$, 并假设 (P, V) 中验证者首先执行。^② 用 $\ell = \ell(|x|)$ 表示 V 使用的随机数个数, 并假设 $\ell > t$ 。回顾在前面假设 P 是一个优化的证明者 (关于 V), 且 (不失一般性) P 为确定性的。用 $\langle P, V(r) \rangle(x)$ 表示 P 与 V 关于 x 交互过程的完整副本, 其中 V 使用随机数 r , 也就是说, $\langle P, V(r) \rangle(x) = (\alpha_1, \beta_1, \dots, \alpha_t, \beta_t, \sigma)$, 其中 $\sigma = V(x, r, \beta_1, \dots, \beta_t) \in \{0, 1\}$ 是 V 的最终判决, 且对任意 $i = 1, 2, \dots, t$, 有 $\alpha_i = V(x, r, \beta_1, \dots, \beta_{i-1})$ 及 $\beta_i = P(x, \alpha_1, \dots, \alpha_i)$ 。

对任意以 P -消息 $\gamma = (\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1})$ 结束的部分副本, 我们将与部分副本 γ 一致的随机序列记作 $\text{ACC}_x(\gamma)$, 并令 V 与 P 交互后接受 x , 也就是说, $r \in \text{ACC}_x(\gamma)$ 当且仅当对某个 $\gamma' \in \{0, 1\}^{2(t-i) \cdot \text{poly}(|x|)}$, 有 $\langle P, V(r) \rangle(x) = (\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1}, \gamma', 1)$ 。在以 V -消息结束的部分副本中使用同样的记号, 即 $r \in \text{ACC}_x(\alpha_1, \beta_1, \dots, \alpha_i)$ 当且仅当对某个 γ' , 有 $\langle P, V(r) \rangle(x) = (\alpha_1, \beta_1, \dots, \alpha_i, \gamma', 1)$ 。

动机

通过适当的错误缩减, 可以假设 (P, V) 的合理性误差 $\mu = \mu(|x|)$ 小于 $\text{poly}(\ell)^{-t}$ 。因此, 对任意“是”实例 x , 有 $|\text{ACC}_x(\lambda)| = 2^\ell$, 而对任意“否”实例, 有 $|\text{ACC}_x(\lambda)| \leq \mu \cdot 2^\ell$ 。事实上, 初始集合的规模缝隙是巨大的, 且可在相应的剩余集合 (即 $\text{ACC}_x(\alpha_1, \beta_1, \dots, \alpha_i)$) 规模间保持一个缝隙, 而每轮最多丢失一个 $\text{poly}(\ell)$ 因子。关键问题是, 对任意的部分副本 $\gamma = (\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1})$, 有

① 我们指出, 上述协议只是构造 6.32 思想的几种实现方法之一。首先, 注意到, 另一种实现可能将选择 Hash 的任务交给验证者, 而验证者随机选择一个函数。虽然这似乎更自然, 但实际上对于“类合理性”性质 (即性质二) 并无益处。另外, 此时, 验证者选择的 Hash 函数可能 (可能性极小) 并非良好的, 从而使性质一不满足 (即要求输出总是属于 S)。其次, 回顾在上述协议中, 最后一步是证明者随机选择 S 中位于 (由验证者) 所选单元中的一个元素。另一种方法将这一步替换为两步, 第一步中证明者发送该单元中 (S 的) $(1 - \varepsilon) \cdot N / 2^m$ 个元素的列表, 然后验证者输出在该表中均匀选择的元素。这种方法可以改善“类合理性”性质 (即验证者的输出属于 S' 的概率至多为 $(|S'| / N) + \varepsilon$), 但是要求额外的消息 (这是我们希望避免的, 虽然并不重要)。

② 注意: 如果证明者在 (P, V) 中先执行, 则对其第一个消息的模仿将无需任何代价 (不使轮数增加)。

$$|\text{ACC}_x(\gamma)| = \sum_{\alpha} |\text{ACC}_x(\gamma, \alpha)| \quad (\text{F.5})$$

而 $|\text{ACC}_x(\gamma, \alpha)| = \max_{\beta} \{|\text{ACC}_x(\gamma, \alpha, \beta)|\}$ 。显然, 利用足够的 β 可以证明 $|\text{ACC}_x(\gamma, \alpha)|$ 较大, 还可证明 $|\text{ACC}_x(\gamma, \alpha, \beta)|$ 也较大。类似地, 证明 $|\text{ACC}_x(\gamma)|$ 较大可归约为证明和 $\sum_{\alpha} |\text{ACC}_x(\gamma, \alpha)|$ 较大。问题是这个和中可能包含指数多项, 这使我们甚至无法读出每一项的值。^①如附录 F.2.1.1 所暗示的, 可将这些项聚合为 l 个簇, 使得第 j 个簇由势约为 2^j 的集合构成 (即满足 $2^j \leq |\text{ACC}_x(\gamma, \alpha)| < 2^{j+1}$ 的 α)。

这些簇中必一个包含 $|\text{ACC}_x(\gamma)|$ 规模中 $1/2\ell$ 的部分, 从而可主要讨论该簇, 也就是说, 我们构造的证明者可以识别一个适当的 j (即在第 j 个簇的集合中至少有 $|\text{ACC}_x(\gamma)|/O(\ell)$ 个元素), 并证明至少存在 $N = |\text{ACC}_x(\gamma)|/(2\ell \cdot 2^{j+1})$ 个集合 (即 $\text{ACC}_x(\gamma, \alpha)$), 每一个规模不小于 2^j 。注意: 这样就证明了 $|\text{ACC}_x(\gamma)|$ 大于 $N \cdot 2^j = |\text{ACC}_x(\gamma)|/O(\ell)$, 这意味着在 $\text{ACC}_x(\gamma)$ 规模中已经丢失了一个 $O(\ell)$ 因子。但是如前所述, 这种损失是可以接受的。

在讨论实际协议之前, 首先要提出一种方法来证明至少存在 N 个集合 (即 $\text{ACC}_x(\gamma, \alpha)$), 每一个规模不小于 2^j 。可利用随机选择协议证明 (同时令规模参数为 N), 目标是选择一个集合 (或其索引 α)。如果真的存在 N 个这样的集合, 则协议的第一个性质保证了这样的集合总能被选中, 从而可利用选中的规模不小于 2^j 的集合进行下一次迭代 (并可建立相应的下界)。因此, 在当前迭代中输入一个有效断言, 便可得到一个新的断言, 并进入下一次迭代。另一方面, 假设 $|\text{ACC}_x(\gamma)| \ll N \cdot 2^j$, 则协议的第二个性质蕴含了^②, 以不小于 $1 - (1/3t)$ 的概率, 选中的 α 满足 $|\text{ACC}_x(\gamma, \alpha)| < \text{poly}(\ell) \cdot |\text{ACC}_x(\gamma)|/N \ll 2^j$, 而在下一次迭代中, 需要证明选取中的集合规模不小于 2^j 。从而以不小于 $1 - (1/3t)$ 的概率向当前迭代输入一个错误断言时, 设其错误因子为 $F \gg \text{poly}(\ell)$, 则在进入下一轮迭代时, 提出错误断言的因子至少是 $F/\text{poly}(\ell)$ 。

要注意的是, 虽然上述动机讨论涉及到对各种集合规模证明下限, 但实际实现时却要在这些集合中随机选择元素。如果集合规模比断言中小, 则选中的集合很有可能在这些集合之外, 从而最终会被检测到。

构造 F.4 (实际的协议) 对于公共输入 x , P 与 V 之间包含 $2t$ 个消息的交互可通过 t 次迭代被“拟模仿”, 其中 $t = t(|x|)$ 。第 i 次迭代从一个部分副本 $\gamma_{i-1} = (\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1})$ 及断言界限 M_{i-1} 开始, 而在第一次迭代中, 副本 γ_0 为空串, 且 $M_0 = 2^\ell$ 。第 i 次迭代过程如下。

(1) 证明者确定一个索引 j , 使得簇 $C_j = \{\alpha: 2^j \leq |\text{ACC}_x(\gamma_{i-1}, \alpha)| < 2^{j+1}\}$ 的规模至少为 $N \stackrel{\text{def}}{=} M_{i-1}/(2^{j+2}\ell)$, 将 j 发送给验证者。注意: 如果 $|\text{ACC}_x(\gamma_{i-1})| \geq M_{i-1}$, 则这样的 j 一定存在。

(2) 证明者对规模参数 N 调用随机选择协议, 选中 $\alpha \in C_j$, 这里为了简单, 我们假设 $C_j \subseteq \{0, 1\}^\ell$ 。回顾这个公开掷硬币协议包含 3 个消息, 其中第一个和最后一个消息由证明者

① 此外, 我们无法验证关于这些值中多于一个的断言, 因为每轮中至少检查两个值将使计算代价呈指数增长 (即时间复杂度成为轮数的指数函数)。

② 对于损耗系数 $L = \text{poly}(\ell)$, 考虑集合 $S' = \{\alpha: |\text{ACC}_x(\gamma, \alpha)| \geq L \cdot |\text{ACC}_x(\gamma)|/N\}$, 则 $|S'| \leq N/L$, 从而, S' 中一个元素被选中的概率至多为 $\text{poly}(\ell)/L$, 当使用适当选择的 L 时, 其上限为 $1/3t$ 。

发出。将协议的输出记作 α_i 。

(3) 证明者确定 β_i , 满足 $\text{ACC}_x(\gamma_{i-1}, \alpha_i, \beta_i) = \text{ACC}_x(\gamma_{i-1}, \alpha_i)$, 并将 β_i 发送给验证者。

在进入下一次迭代前, 令 $M_i \leftarrow 2^j$ 及 $\gamma_i = (\alpha_1, \beta_1, \dots, \alpha_i, \beta_i) \equiv (\gamma_{i-1}, \alpha_i, \beta_i)$ 。

最后一次迭代完成后, ^①证明者对规模参数 $N = M_t$ 调用随机选择协议, 以选出 $r \in \text{ACC}_x(\alpha_1, \beta_1, \dots, \alpha_t, \beta_t)$ 。在得到这个 r 之后, 验证者接受输入当且仅当 $V(x, r, \beta_1, \beta_2, \dots, \beta_t) = 1$ 且对任意的 $i = 1, 2, \dots, t$, 有 $\alpha_i = V(x, r, \beta_1, \dots, \beta_{i-1}) = 1$, 其中 α_i 和 β_i 在上述迭代中确定。

注意: 每次迭代中的 3 个步骤都涉及到 (公开掷硬币) 验证者的单个消息, 从而上述协议在实现时共有 $2t + 3$ 个消息。

显然, 如果 x 为“是”实例, 则证明者令验证者以概率 1 接受 (因为每次迭代中存在足够大的簇, 且随机选择协议保证了选中 α_i 的属于该簇)。^②另一方面, 如果 x 为“否”实例, 则利用 (P, V) 的低合理性误差, 可以证明构造 F.4 的合理性。这在下述断言中证明, 其中涉及到一个足够大的多项式 p 。

命题 F.5 假设 $|\text{ACC}_x(\lambda)| < \delta^{t+1} \cdot 2^\ell$, 其中 $\delta = 1/p(\ell)$, 则构造 F.4 中的验证者接受 x 的概率小于 $1/2$ 。

证明概要: 首先证明, 对任意 $i = 1, 2, \dots, t$, 如果 $|\text{ACC}_x(\gamma_{i-1})| < \delta^{t+1-(i-1)} \cdot M_{i-1}$, 则 $|\text{ACC}_x(\gamma_i)| < \delta^{t+1-i} \cdot M_i$ 以至少 $1 - (1/3t)$ 的概率成立。给定任意 i , 令 j 为第 i 次迭代中证明者在第一步选择的值, 并定义 $S' = \{\alpha : |\text{ACC}_x(\gamma_{i-1}, \alpha)| \geq \delta^{t+1-i} \cdot 2^j\}$, 则

$$|S'| \cdot \delta^{t+1-i} 2^j \leq |\text{ACC}_x(\gamma_{i-1})| < \delta^{t+1-(i-1)} \cdot M_{i-1}$$

其中第二个不等式为断言中的假设。令 $N = M_{i-1} / (2^{j+2} \ell)$ (如本次迭代中第一步所做的), 则有 $|S'| < 4\ell \delta \cdot N$ 。由随机选择协议的第二个性质 (本次迭代的第二步中关于规模参数 N 调用), 有

$$\Pr[\alpha_i \in S'] \leq \text{poly}(\ell) \cdot \frac{|S'|}{N} \leq \text{poly}(\ell) \cdot \delta$$

这个值小于 $1/3t$ (假设确定的多项式 p 足够大)。因此, $|\text{ACC}_x(\gamma_{i-1}, \alpha_i)| < \delta^{t+1-i} \cdot 2^j$ 成立的概率不小于 $1 - (1/3t)$ 。利用 $M_i = 2^j$ (第三步中) 及对任意 β , 有 $|\text{ACC}_x(\gamma_{i-1}, \alpha_i, \beta)| < |\text{ACC}_x(\gamma_{i-1}, \alpha_i)|$, 可以证明关于 $|\text{ACC}_x(\gamma_i)|$ 的断言。

利用假设 $|\text{ACC}_x(\gamma_0)| < \delta^{t+1} \cdot M_0$ 及上述断言, 可以证明, 上述 t 次迭代的执行过程中, 以至少 $2/3$ 的概率产生 γ_t 和 M_t , 且满足 $|\text{ACC}_x(\gamma_t)| < \delta \cdot M_t$ 。此时, 最后一次调用随机选择协议 (关于规模参数 M_t) 时, 以至多 $\text{poly}(\ell) \cdot \delta < 1/6$ 的概率生成 $\text{ACC}_x(\gamma_t)$ 中元素, 否则, 验证者将拒绝 (因为验证者检查关于随机选择协议的输出 r 的条件在逻辑上等价于 $r \in \text{ACC}_x(\gamma_t)$)。命题得证。 $\square$

F.2.2 $\mathcal{AM}$ 的线性加速

本节证明定理 F.3。我们的证明不同于 Babai 和 Moran^[23] 的原始证明, 主要区别是对基本

① 作为替代, 可以修改 (P, V) , 向其中加入最后一个 V -消息, 其中包含 V 的内部随机数 (即 r)。此时, 对随机选择协议的额外调用可作为一种特例, 用于处理增加的第 $t+1$ 次迭代。

② 因此, 在最后一次调用随机选择协议时, 验证者总能得到 $r \in \text{ACC}_x(\gamma_t)$, 并接受。

转换（从 MA 到 AM）的分析。

我们采用公开掷硬币（即 Arthur-Merlin）交互式证明的标准术语，其中验证者被称为 Arthur，证明者被称为 Merlin。更重要的是，将这样一个证明系统对任意给定公开输入 x 的执行过程，看作是在诚实的 Arthur 和强大的 Merlin 间进行的一个（全信息）游戏（由 x 标记）。这些参与方轮流行动，其中 Arthur 的行为是随机的，而 Merlin 的行为关于某个给定的（多项式时间内计算的）谓词 v_x 是最优的，该谓词是根据游戏执行的完全副本而计算的。我们强调（与一般的交互式证明相反），Arthur 的行为在可能值的集合上均匀分布，这些值是独立于以前的行为（即集合 $\{0,1\}^{l(|x|)}$ ）而预先计算的。游戏的值定义为一次执行的期望值，该期望根据 Arthur 的行为求出（并假设 Merlin 的行为是最优的）。

不失一般性，可以假设 Arthur 的所有消息具有相同长度，记作 $\ell = \ell(|x|)$ 。类似地，Merlin 的消息长度为 $m = m(|x|)$ 。

回顾 $\mathcal{AM} = \mathcal{AM}(2)$ 表示一个两消息系统，其中 Arthur 首先行动，在收到 Merlin 的应答后并不生成随机数，而 $\mathcal{MA} = \mathcal{AM}(1)$ 表示一个单消息系统，其中 Merlin 发送一个消息，Arthur 收到该消息后生成随机数。因此，可将 $\mathcal{AM}$ 和 $\mathcal{MA}$ 均看作是两阶段游戏，其中两个参与方的行为顺序不同。很快将看到（在附录 F.2.2.1 中），“MA 顺序”可以用“AM 顺序”来模拟（即 $\mathcal{MA} \subseteq \mathcal{AM}$ ）。这一事实是“圈加速”转换的基础（见附录 F.2.2.2）。

F.2.2.1 基本转换（从 MA 到 AM）

基本思想是将一个 MA-游戏（即一个两阶段游戏，其中 Merlin 先行动，然后 Arthur 行动）转化为一个 AM-游戏（Arthur 先行动，然后才是 Merlin）。在原始游戏中（对于输入 x ），Merlin 首先发送消息 $\beta \in \{0,1\}^m$ ，然后 Arthur 以随机的 $\alpha \in \{0,1\}^\ell$ 响应，而 Arthur 的判决（即游戏执行的值）由 $v_x(\beta, \alpha) \in \{0,1\}$ 给出。在新游戏（图 F.1）中，行动的顺序将交换，但为了限制交换可能带给 Merlin 的收获，要求其提供一个应答，能“符合”Arthur 的几个随机消息。也就是说，对于要规定的参数 t ，Arthur 首先发送随机序列 $(\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)}) \in \{0,1\}^{t \cdot \ell}$ ，然后 Merlin 以字符串 $\beta \in \{0,1\}^m$ 响应，当且仅当对任意 $i=1, 2, \dots, t$ ， $v_x(\beta, \alpha^{(i)})=1$ 时（即新游戏副本的值被定义为 $\prod_{i=1}^t V_x(\beta, \alpha^{(i)})$ 时），Arthur 才接受。直观上，Merlin 获得的优势是在看到 Arthur 的行为之后才选择自己的行为，但 Merlin 的选择必须符合 Arthur 的 t 个行为，这就使 Merlin 的收获极有限（如果 t 足够大）。



图 F.1 MA-游戏向 AM-游戏的转换。原始 MA-游戏的副本 (β, α) 的值由 $V_x(\beta, \alpha)$ 给出，而新游戏副本

$((\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)}), \beta)$ 的值由 $\prod_{i=1}^t V_x(\beta, \alpha^{(i)})$ 给出

回顾新游戏副本的 $(\bar{\alpha}, \beta)$ 值 v_x' , 其中 $\bar{\alpha} = (\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)})$ 被定义为 $\prod_{i=1}^t V_x(\beta, \alpha^{(i)})$ 。因此, 新游戏的值定义为

$$E_{\bar{\alpha}} \left[\max_{\beta} \left\{ \prod_{i=1}^t v_x(\beta, \alpha^{(i)}) \right\} \right] \quad (\text{F.6})$$

其上界为

$$E_{\bar{\alpha}} \left[\max_{\beta} \left\{ \frac{1}{t} \sum_{i=1}^t v_x(\beta, \alpha^{(i)}) \right\} \right] \quad (\text{F.7})$$

注意: 当原始 MA-游戏的值为 1 (即 x 为“是”实例) 时, 式 (F.7) 提供的上界是紧凑的, 此时, 新游戏的值也是 1 (因为此时存在一个行为 β , 使得对任意 α , 有 $v_x(\beta, \alpha) = 1$)。然而, 有趣的是, 使 Merlin 能有所收获的交流中, 原始 MA-游戏的值一定远远小于 1 (即 x 为“否”实例)。一个主要论断是, 对于适当选择的 t , 使 Merlin 从交换中收获良多是不太可能的。回顾在原始的 MA-游戏中, Merlin 在选择 β 时并未考虑 Arthur 选择的 α , 从而 Merlin 的“收益” (即游戏的值) 可表示为 $\max_{\beta} \{E_{\alpha}(v_x(\beta, \alpha))\}$ 。在新的 AM-游戏中, Merlin 基于 Arthur 选择的序列 $\bar{\alpha}$ 来选择 β , 我们在式 (F.7) 中已经给出了其 (在新的 AM-游戏中) “收益”的上限。因此, Merlin 从交换中的收获其实是额外收益 (将新的 AM-游戏与原先的 MA-游戏相比)。还可计算 Merlin 的收获超过某个参数 (记作 δ) 的概率上限, 即

$$\begin{aligned} p_{x,\delta} &\stackrel{\text{def}}{=} \Pr_{(\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)})} \left[\max_{\beta} \left\{ \frac{1}{t} \cdot \sum_{i=1}^t v_x(\beta, \alpha^{(i)}) \right\} > \max_{\beta} \{E_{\alpha}(v_x(\beta, \alpha))\} + \delta \right] \\ &\leq \Pr_{(\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)})} \left[\exists \beta \in \{0, 1\}^m \text{ s.t. } \left| \frac{1}{t} \cdot \sum_{i=1}^t v_x(\beta, \alpha^{(i)}) - E_{\alpha}(v_x(\beta, \alpha)) \right| > \delta \right] \\ &\leq 2^m \cdot \exp(-\Omega(\delta^2 \cdot t)) \end{aligned}$$

其中最后一个不等式是结合了联合界限与 Chernoff 界而得到的。将原始游戏的值记作 $V_x = \max_{\beta} \{E_{\alpha}(v_x(\beta, \alpha))\}$, 可求出式 (F.7) 中的上限为 $p_{x,\delta} + V_x + \delta$ 。利用 $t = O((m+k)/\delta^2)$, 有 $p_{x,\delta} \leq 2^{-k}$, 因此, 有

$$V_x' \stackrel{\text{def}}{=} E_{\bar{\alpha}} \left[\max_{\beta} \left\{ \frac{1}{t} \sum_{i=1}^t v_x(\beta, \alpha^{(i)}) \right\} \right] \leq \max_{\beta} \{E_{\alpha} v_x((\beta, \alpha))\} + \delta + 2^{-k} \quad (\text{F.8})$$

显然, 式 (F.7) 的下界是 V_x (因为 Merlin 只需利用 MA-游戏中的最优行为)。特别地, 利用 $\delta = 2^{-k} = 1/8$, 并假设 $V_x \leq 1/4$ 时, 可以得到 $V_x' < 1/2$ 。因此, 从对某个集合的 MA 证明系统出发, 可以得到对同一集合的 AM 证明系统, 从而证明了 $\mathcal{MA} \subseteq \mathcal{AM}$ 。

扩展

注意到, 上述转换及其分析中没有提及一个事实, 即 $v_x(\beta, \alpha)$ 足以从 (β, α) 中计算。进一步地, 对任意的 $v_x(\cdot, \cdot) \in [0, 1]$, 该分析仍有效, 因为对任意 $v_1, v_2, \dots, v_t \in [0, 1]$, 有 $\prod_{i=1}^t v_i \leq \left(\prod_{i=1}^t v_i \right)^{1/t} \leq \sum_{i=1}^t v_i / t$ 。因此, 我们可以将上述转换应用于公开掷硬币证明系统中任意两个连续的 Merlin-Arthur 行为中, 条件是所有连续行为都在 t 个副本中执行, 每个副本对应于交换中使用的不同 $\alpha^{(i)}$ 。也就是说, 如果第 j 个行为由 Merlin 做出, 则可以交换第 j

和第 $j+1$ 次行为的参与方, 即令 Arthur 进行第 j 次行为, 在 Merlin 的应答 β 之后发送 $(\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)})$ 。后续行为将在 t 个副本中执行, 其中第 i 个副本对应于行为 $\alpha^{(i)}$ 和 β 。新游戏的值至多增加 $2^{-k} + \delta < 1/4$, 从而通过交换两个步骤得到一个“等价的”游戏。示意性地, 在 MA 的中间 (以粗体字表示) 可以用 $[AM]^{j_1} \mathbf{AAM} [MA]^{j_2}$ 代替 $[AM]^{j_1} \mathbf{AMA} [MA]^{j_2}$, 这又允许两个相继的 A-行为 (以及 $j_2 \geq 1$ 时两个相继的 M-行为) 的衰退。特别地 (只在 $j_1 = 0$ 情形中使用), 可以得到 $A[MA]^{j+1} = A[MA]^j = \dots = \mathbf{AMA} = \mathbf{AM}$ 。因此, 对任意常数 f , 有 $\mathcal{AM}(f) = \mathcal{AM}(2)$ 。

我们强调, 上述交换过程只能应用常数次, 因为每次交换时, 消息长度会增加一个 $t = \Omega(m)$ 因子。因此, 要求用不同方法处理非常数数量的消息 (即无界限函数 f)。

F.2.2.2 增强型转化 (从 $[MAMA]^j$ 到 $[AMA]^j A$)

持续应用“MA 到 AM 交换”可以将轮数减少一个常数。然而, 每次应用交换时, 所有相继行为都要 (并行地) 执行 t 次。也就是说, “MA 到 AM 交换”将游戏的剩余部分分成 t 个独立副本, 从而交换过程不能执行多于常数次。幸运的是, 式 (F.7) 提出一种方法, 可以将游戏缩小成一个副本: 只需令 Arthur 均匀选择 $i \in [t]$, 并让各参与方继续执行第 i 个副本。^① 为了避免引入一个 Arthur-Merlin 交替, Arthur 的额外行为被延迟至 Merlin 的行为之后 (图 F.2)。示意性地 (将行为用粗体字表示), 将 **MAMA** 用 **AMMAA=AMA** 代替 (而不是用 **AMAMA** 代替 **MAMA**, 这并不减少行为的交替次数)。

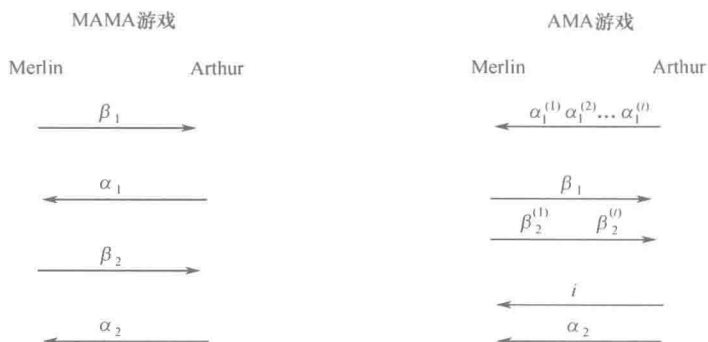


图 F.2 MAMA 到 AMA 的转化。原始 MAMA-游戏的副本 $(\beta_1, \alpha_1, \beta_2, \alpha_2)$ 的值由 $v_x(\beta_1, \alpha_1, \beta_2, \alpha_2)$ 给出, 而新的 AMA-游戏的副本 $((\alpha_1^{(1)}, \alpha_1^{(2)}, \dots, \alpha_1^{(t)}), (\beta_1, \beta_2^{(1)}, \beta_2^{(2)}, \dots, \beta_2^{(t)}), (i, \alpha_2))$ 的值由 $v_x(\beta_1, \alpha_1^{(i)}, \beta_2^{(i)}, \alpha_2)$ 给出

通过上述扩展交换, 游戏的值可由式 (F.7) 给出, 可以写成

$$E_{\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(t)}} \left[\max_{\beta} \left\{ E_{i \in [t]} \left(v_x(\beta, \alpha^{(i)}) \right) \right\} \right]$$

其上限为 $\max_{\beta} \{ E_{\alpha} (v_x(\beta, \alpha)) \} + \delta + 2^{-k}$ (见式 (F.8))。与附录 F.2.2.1 相同, 该方法可应用于任意公开掷硬币交互证明中两个相继的 Merlin-Arthur 行为中。回顾前面讲过, 为了避免引入额外的 Arthur 行为, 可将 Arthur 的最后一次行为延迟至 Merlin 的下一行为之后。因此, 可以与任意 4 个连续的、以 Merlin 开始的行为块中的前两个行为应用增强型交换, 将序列 MAMA

① 事实上, 式 (F.7) 的不严格形式在这里起了关键作用 (与式 (F.6) 相比)。

转化为 $AMMAA=AMA$ (图 F.2)。关键在于, 在该行为块之后的行为仍保持不变。因此, 可以将增强型的“MA 到 MA 交换”(实际上是“MAMA 到 AMA 交换”)同时应用于游戏中不相交的部分。示意性地, 可以用 $[AMA]^j = A[MA]^j$ 代替 $[MAMA]^j$ 。注意: Merlin 在每次交换中的收益上限为 $\delta + 2^{-k}$, 但是如果选择 $t = \tilde{O}\left(f(|x|)\right)^2 \cdot m(|x|) = \text{poly}(|x|)$, 则可将其总收益控制为一个常数 (利用 $\delta = 2^{-k} = 1/8f(|x|)$)。因此, 就证明了 $\mathcal{AM}(4f) \subseteq \mathcal{AM}(2f+1)$, 定理 F.3 得证。

附录 G

一些计算问题

虽然相比于复杂性类而言，特定计算问题似乎是次要的，但后者还需利用前者来澄清和解释。本附录给出这样一些计算问题的定义，并按照其研究对象（如图、布尔公式等）进行分类。

首先介绍上述计算问题的研究对象在表示中的核心问题。一般原则是，将所有集合的元素“简洁地”表示为二进制串（没有过多冗余）。例如，一个有限集合 S （如图中的顶点集合或出现在布尔公式中的变量集合）中的元素用长度为 $\log_2 |S|$ 的二进制字符串表示。

G.1 图

图论早已被视为计算机科学的学生要掌握的众多有用的数学分支之一。计算机科学的研究方法自然是算法；本书的兴趣并不在存在性证明或计算技巧上，而在于寻找解决相关问题的高效算法，或者证明这样的算法不存在。算法图论最早由欧拉开创，但其在近十年的发展是令人激动的，也是革命性的。

西蒙·埃文，图论算法^[71]

一个简单图 $G=(V, E)$ 由顶点的有限集 V 和边的有限集 E 组成，其中每条边是顶点的一个无序对；即 $E \subseteq \{\{u, v\} : u, v \in V \wedge u \neq v\}$ 。简单图中不包含自环和平行边，但在普通图（即非简单图）中是允许的，一般图中的 E 是一个多重集，其中可能（除 V 中顶点构成的子集外）还包含自环。顶点 u 被称为边 $\{u, v\}$ 的一个端点，而边 $\{u, v\}$ 被称为依附于 v 。此时，称 u 和 v 在图中是相邻的， u 是 v 的邻居。在图 G 中依附于一个顶点的边的数量被定义为该顶点的度。

再考虑图上的各种子结构，最简单的一种是路径。在图 $G=(V, E)$ 中，路径是顶点 $(v_0, v_1, \dots, v_l)$ 的序列，使得对于任意 $i \in [l] \stackrel{\text{def}}{=} \{1, 2, \dots, l\}$ ， v_{i-1} 和 v_i 在 G 中是相邻的。此时，称路径具有长度 ℓ 。简单路径是指在一条路径中每个顶点最多出现一次，这意味着在 G 中简单路径的长度不会超过 $|V|-1$ 。如果图中每一对顶点之间都存在路径，则称该图为连通图。

环是最后一个顶点等于第一个顶点的路径（即 $v_l = v_0$ ）。在环 $(v_0, v_1, \dots, v_l)$ 中，如果 $l > 2$ 且 $|\{v_0, v_1, \dots, v_l\}| = l$ ，则该环是一个简单环（例如，如果 $v_i = v_j$ ，那么 $i \equiv j \pmod{\ell}$ ），并且环 (u, v, u)

不是一个简单环)。如果一个图没有简单环,则称该图为非循环图(或者森林),如果它还是连通的,就称其为树。注意:图 $G=(V,E)$ 是一棵树当且仅当它是连通的,且 $|E|=|V|-1$, 树中每一对顶点之间只有唯一的简单路径。

图 $G=(V,E)$ 的子图是任意满足 $V' \subseteq V$ 和 $E' \subseteq E$ 的图 $G'=(V',E')$ 。注意: G 中的简单环是 G 的一个连通子图,其中每个顶点的度恰为 2。图 $G=(V,E)$ 的一个导出子图(Induced Subgraph)是一个子图 $G'=(V',E')$,其中包含了顶点属于 V' 的 E 中所有边。此时称 G' 是由 V' 导出的子图。

有向图

再来考虑(简单)有向图,其中的边是顶点的有序对。在这种情况下,边的集合是 $V \times V \setminus \{(v,v): v \in V\}$ 的一个子集,并且边 (u,v) 和边 (v,u) 被称为反向平行。一般有向图的定义(如非简单的)是类似的。边 (u,v) 被视为从 u 到 v , 因此称为 u 的外出边(分别地, v 的进入边)。顶点的外出边(或进入边)的数量称为该顶点的出度(或入度)。有向路径和其他相关对象的定义类似;举个例子,设 $v_0, v_1, \dots, v_l$ 为一条路径,如果对于任意 $i \in [l]$, (v_{i-1}, v_i) 都是有向边(该路径从 v_{i-1} 到 v_i 是有方向性的),则称 $v_0, v_1, \dots, v_l$ 为一条有向路径。通常把一对反向平行的边视为一个简单有向环。

有向无环图(DAG)是不含有向环的有向图。每一个 DAG 至少有一个顶点出度(或入度)为零,被称作汇点(或源点)。在一个简单定向非循环图 $G=(V,E)$ 中,如果 $|E|=|V|-1$ 且存在唯一一个出度(或入度)为零的顶点,则称该图为一个内向(或外向)的有向树,其中出度(或入度)为零的顶点被称为根。注意,在内向(外向)有向树中每个顶点可通过唯一的有向路径^①到达根(或从根可到达该顶点)。

表示

通常图由其邻接矩阵和/或其依附列表来表示。简单图 $G=(V,E)$ 的邻接矩阵是一个 $|V| \times |V|$ 的布尔矩阵,在该矩阵中,当且仅当 G 中 i 和 j 是相邻结点时第 (i,j) 项为 1。 G 的依附列表由 $|V|$ 个序列组成,其中第 i 个序列是依附于顶点 i 的边集的有序列表。

计算问题

图上的简单计算问题包括判定给的图是否为连通(和/或非循环)图以及在给定的图中寻找最短路径。其他简单问题还有判定给定的图是否为二分图,如果图 $G=(V,E)$ 存在一种着色方案,可以为相邻顶点分配不同颜色,则称该图是二分图(或 2-可着色图)。这些问题都易通过广度优先搜索来解决。

再看更复杂的但能在多项式时间内解决的任务,我们会提及在给定图中找到一个完全匹配(或者一个最大匹配)的问题,当所有顶点的度为 1 时,一个匹配就是一个子图,完全匹配是包含图中所有顶点的匹配,最大匹配是包含顶点数最多(在该图的所有匹配中)的匹配。

图上的困难问题包括判定给定图是否为 3-可着色的(即 G3C),这是一个 NP-完全问题。还有其他一些 NP 完全问题。

- 图 $G=(V,E)$ 的**汉密尔顿路径**(或汉密尔顿回路),是一个通过 G 的所有顶点的简单路径(回路)。这样的路径(回路)的长度为 $|V|-1$ ($|V|$)。问题是判定给定的图是否包含汉密尔顿路径(回路)。

^① 注意:在任意 DAG 中,存在从每个顶点 v 到一些汇点的有向路径(从一些源点到每个顶点)。在内向(外向)有向树中,这个汇点(源点)必须是唯一的。条件 $|E|=|V|-1$ 强调了路径的唯一性,因为(结合相关条件)这意味着当前的图(可通过忽略边的方向性来得到)是一棵树。

• 图 $G=(V,E)$ 的**独立集** (或团) 是一组顶点的集合 $V' \subseteq V$, 使得由 V' 导出的子图中不含任何边 (包含所有边)。问题是判定给定的图是否有给定规模的独立集 (或团)。

• 图 $G=(V,E)$ 的**顶点覆盖** 是一组顶点的集合 $V' \subseteq V$, 使得 E 中的每一个边至少有一端在 V' 中。注意: 当且仅当 $V \setminus V'$ 是 V 的独立集时, V' 是 G 的顶点覆盖。

一个被认为既非 P 类也非 NP-完全的自然计算问题是**图的同构**问题。问题的输入是两个图, $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$, 问题是当且仅当 $\{\phi(u), \phi(v)\}$ 在 E_2 中时, 是否存在一个 1-1 满射 $\phi: V_1 \rightarrow V_2$, 使得 $\{u,v\}$ 在 E_1 中。(这样的映射被称为同构。)

G.2 布尔公式

在 1.2.4.3 节中, 将布尔公式定义为布尔电路 (1.2.4.1 节) 的一种特例。这里用更传统的方式, 将布尔公式定义为由变量名和各种连接词组成的字母表上的结构化序列。布尔公式最方便的定义采用如下的递归方式:

- 某个变量是一个布尔公式。
- 如果 $\phi_1, \phi_2, \dots, \phi_t$ 是布尔公式并且 ψ 是一个 t 进制布尔操作, 那么, $\psi(\phi_1, \phi_2, \dots, \phi_t)$ 是一个布尔公式。

通常考虑 3 种布尔操作: 否定的一元运算 (记为 neg 或者 $\neg$)、(有界或无界) 合取和析取 (分别记为 $\wedge$ 和 $\vee$)。另外, 约定一些速记方式, 用 $\neg\phi$ 表示 $\neg(\phi)$, 用 $(\wedge_{i=1}^t \phi_i)$ 或者 $(\phi_1 \wedge \dots \wedge \phi_t)$ 代替 $\wedge(\phi_1, \phi_2, \dots, \phi_t)$, 对于 $\vee$ 也是类似。

两个重要的布尔公式是 CNF 和 DNF。CNF 是对变量和/或其否定的析取的合取; 也就是说, 如果每个 ϕ_i 形如 $(\vee_{j=1}^{t_i} \phi_{i,j})$ 则 $\wedge_{i=1}^t \phi_i$ 是一个 CNF, 其中每个 $\phi_{i,j}$ 是一个变量或变量的否定 (被称为文字)。如果对于每个 i 有 $t_i \leq 3$, 则称该公式是一个 3CNF。同样, DNF 公式定义为对文字的合取的析取。

布尔公式在对其变量的真值赋值的下的取值, 是根据其结构递归地定义。例如, 当且仅当每个 ϕ_i 在 τ 下的值为真时, $\wedge_{i=1}^t \phi_i$ 在赋值 τ 下的值为真。如果存在一种变量的真值赋值 τ , 使得 ϕ 在 τ 下的值为真, 称公式 ϕ 是可满足的。

可满足的 CNF (3CNF) 公式的集合用 SAT (3SAT) 来表示, 其成员判定是一个 NP-完全问题。即使限制到 3 DNF 公式, 重言式集合 (即在任意赋值的值都为真的公式) 也是 coNP-完全的。

量化布尔公式

与上述非量化布尔公式不同, 量化布尔公式中为每个变量增加了一个量词。也就是说, 如果 ϕ 是关于变量 $x_1, x_2, \dots, x_n$ 的布尔公式, $Q_1, Q_2, \dots, Q_n$ 是布尔量词 (即每个 Q_i 为 $\exists$ 或 $\forall$), 则 $Q_1 x_1 \dots Q_n x_n \phi(x_1, x_2, \dots, x_n)$ 是量化布尔公式。 k -交替的量化布尔公式是这样一种量化布尔公式, 从一个存在量词开始, 在存在量词与全称量词之间进行 k 次转换。例如, $\exists x_1 \exists x_2 \forall x_3 \phi(x_1, x_2, x_3)$ 是一个 2-交替的量化布尔公式 (如果一个量化布尔公式取值为真, 则称其为可满足的)。

可满足 k -交替量化布尔公式的集合表示为 kQBF, 是一个 Σ_k 完全集, 而全体可满足布尔公式的集合记作 QBF, 是 PSPACE-完全的。

前述定义涉及到量化布尔公式的规范形式, 其中所有的量词出现在公式最左边。在一个

更普遍的定义中,公式中每个变量可在任意位置量化(如 $(\forall x_1)(\exists x_2)(x_1 = x_2) \wedge (\exists x_3)(x_3 = x_1)$)。注意:这种推广的公式(被用在定理 5.15 和定理 9.4 的证明中)可以通过将所有的量化“推”到公式左边来转化为规范形式(如 $\forall x_1 \exists x_2 \exists x_3 ((x_1 = x_2) \wedge (x_3 = x_1))$)。

G.3 有限域、多项式及向量空间

各种代数对象、计算问题及技术在复杂性理论中起着重要的作用。最主要的代数对象是有限域、向量空间和多项式。

有限域

用 $GF(q)$ 表示 q 个元素的有限域, q 可能是素数或素数幂。在第一种情况下, $GF(q)$ 包括元素 $\{0, 1, \dots, q-1\}$, 其上运算为模 q 的加法和乘法。 $GF(2)$ 是一个重要特例。在 $q = p^e$ 的情况下, 其中 p 为素数且 $e > 1$, $GF(p^e)$ 的标准表示涉及到 $GF(p)$ 上的 e 次不可约多项式。找到这样的多项式是一个非平凡的计算问题(见参考文献[24])。幸运的是, 对于 $e = 2 \cdot 3^e$ (e 是整数), 多项式 $x^e + x^{e/2} + 1$ 在 $GF(2)$ 上不可约, 这意味着此时求 $GF(2^e)$ 的一种表示是容易的。在一般情况下, 如果 f 是 $GF(p)$ 上的 e 次不可约多项式, 那么, $GF(p^e)$ 可以表示为 $GF(p)$ 上次数至多为 $e-1$ 的多项式集合, 其上运算为模 f 的加法和乘法。

多项式及向量空间

有限域 F (基数至少为 d) 上的 $d-1$ 次多项式集合构成 F 上的 d 维向量空间(如考虑基 $\{1, x, \dots, x^{d-1}\}$)。事实上, 这个向量空间的标准表示涉及到基 $1, x, \dots, x^{d-1}$, 并且多项式 $\sum_{i=0}^{d-1} c_i x^i$ 可以表示为向量 $(c_0, c_1, \dots, c_{d-1})$ 。通过计算 d 个不同的点 $\alpha_1, \alpha_2, \dots, \alpha_d \in F$, 可以得到另一种基。也就是说, $d-1$ 次多项式 p 可以重新表示为 $(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_d))$ 的值序列。当然, 从一种表示到另一种表示需要进行线性变换。也就是说, 对于 $p(x) = \sum_{i=0}^{d-1} c_i x^i$, 有

$$\begin{pmatrix} p(\alpha_1) \\ p(\alpha_2) \\ \vdots \\ p(\alpha_d) \end{pmatrix} = \begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{d-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_d & \cdots & \alpha_d^{d-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{d-1} \end{pmatrix} \quad (G.1)$$

式 (G.1) 中的 (满秩) 矩阵称为范德蒙矩阵。

G.4 矩阵的行列式与常值

$n \times n$ 矩阵 $M = (a_{i,j})$ 的常值定义为集合 $\{1, 2, \dots, n\}$ 的所有排列上的和 $\sum_{\pi} \prod_{i=1}^n a_{i,\pi(i)}$ 。这与行列式的定义相关, 该定义中使用了除一些被否定的元素外之外的同样的和, 即 $M = (a_{i,j})$ 的行列式定义为 $\sum_{\pi} (-1)^{\sigma(\pi)} \prod_{i=1}^n a_{i,\pi(i)}$, 其中如果 π 是一个偶排列 (即能用偶数数量的对换来表示), 则 $\sigma(\pi) = 1$, 否则, $\sigma(\pi) = -1$ 。

相应计算问题 (计算给定矩阵的行列式或常值) 的复杂性似乎大不相同。把矩阵转换成三角形 (即对于每个 $i > j$, $a_{i,j} = 0$) 之后很容易行列式计算, 可以在多项式时间内完成。

相比之下，计算元素在 $\{0,1\}$ 中的矩阵的常值则是 $\#P$ -完全的（见定理 6.20）。

G.5 素数与合数

素数是不能被除自身和 1 之外的任何自然数整除的自然数。不是素数的自然数称为合数，其素因子是能整除它的素数，即如果 $N = \prod_{i=1}^t P_i^{e_i}$ ，其中 P_i 是互不相同的素数(大于 1)且 $e_i \geq 1$ ，那么， s 是 N 的素因子（如果 $t=1$ ，那么， N 是素数幂）。

高斯提出了两个著名的计算问题：素性测试（即给定一个自然数，判定其是素数还是合数）和合数的分解（即给定一个复合数，找到其素因子）。这两种情况下的输入都可以表示为二进制。素性测试属于 P 类（参见参考文献[3]（6.1.1.2 节证明这个问题属于 BPP ）），而人们猜想分解合数是困难的。事实上，许多流行的密码方案都基于这个猜想。

求模平方根

有两个相关的计算问题是在素数和合数模下求（模）平方根。具体来说，**模素数 P 的二次剩余**是一个这样的整数 s ，存在一个数 r 满足 $s \equiv r^2 \pmod{P}$ 。相应的搜索问题（即给定这样的 P 和 s ，寻找 r ）可在概率多项式时间解决（见习题 6.15）。对于合数模，相应的问题在计算上等同于因子分解（见参考文献[183]）；而且，计算模 N 的平方根也可以很容易地归约为分解 N ，分解 N 的问题则可以随机归约为求模 N 的平方根（即使一般情况的意义上）。需要指出，即使是判定给定的整数 s 对于给定的合数模数是否存在平方根，也是一个困难猜想（但尚不知道是否在计算上等价于分解合数）。

参考文献

- [1] S. Aaronson. Complexity Zoo. A continuously updated Web site at http://qwiki.caltech.edu/wiki/Complexity_Zoo/.
- [2] L. M. Adleman and M. Huang. *Primality Testing and Abelian Varieties Over Finite Fields*. Springer-Verlag Lecture Notes in Computer Science (Vol. 1512), 1992. Preliminary version in *19th STOC*, 1987.
- [3] M. Agrawal, N. Kayal, and N. Saxena. PRIMES Is in P. *Annals of Mathematics*, Vol. 160 (2), pages 781–793, 2004.
- [4] M. Ajtai, J. Komlos, and E. Szemerédi. Deterministic Simulation in LogSpace. In *19th ACM Symposium on the Theory of Computing*, pages 132–140, 1987.
- [5] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. In *20th IEEE Symposium on Foundations of Computer Science*, pages 218–223, 1979.
- [6] N. Alon, L. Babai, and A. Itai. A Fast and Simple Randomized Algorithm for the Maximal Independent Set Problem. *J. of Algorithms*, Vol. 7, pages 567–583, 1986.
- [7] N. Alon and R. Boppana. The Monotone Circuit Complexity of Boolean Functions. *Combinatorica*, Vol. 7 (1), pages 1–22, 1987.
- [8] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of Asymptotically Good, Low-Rate Error-Correcting Codes Through Pseudo-Random Graphs. *IEEE Transactions on Information Theory*, Vol. 38, pages 509–516, 1992.
- [9] N. Alon, E. Fischer, I. Newman, and A. Shapira. A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity. In *38th ACM Symposium on the Theory of Computing*, pages 251–260, 2006.
- [10] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple Constructions of Almost k -wise Independent Random Variables. *Journal of Random Structures and Algorithms*, Vol. 3 (3), pages 289–304, 1992. Preliminary version in *31st FOCS*, 1990.
- [11] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 1992. Second edition 2000.
- [12] R. Armoni. On the Derandomization of Space-Bounded Computations. In the proceedings of *Random98*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1518), pages 49–57, 1998.
- [13] S. Arora. Approximation Schemes for NP-Hard Geometric Optimization Problems: A Survey. *Math. Programming*, Vol. 97, pages 43–69, July 2003.
- [14] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, forthcoming.
- [15] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Journal of the ACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.
- [16] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *Journal of the ACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.
- [17] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1998.
- [18] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

- [19] L. Babai. Random Oracles Separate PSPACE from the Polynomial-Time Hierarchy. *Information Processing Letters*, Vol. 26, pages 51–53, 1987.
- [20] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd ACM Symposium on the Theory of Computing*, pages 21–31, 1991.
- [21] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time Has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1 (1), pages 3–40, 1991. Preliminary version in *31st FOCS*, 1990.
- [22] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME Has Publishable Proofs. *Complexity Theory*, Vol. 3, pages 307–318, 1993.
- [23] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes. *Journal of Computer and System Science*, Vol. 36, pages 254–276, 1988.
- [24] E. Bach and J. Shallit. *Algorithmic Number Theory (Volume I: Efficient Algorithms)*. MIT Press, 1996.
- [25] B. Barak. Non-Black-Box Techniques in Cryptography. Ph.D. thesis, Weizmann Institute of Science, 2004.
- [26] W. Baur and V. Strassen. The Complexity of Partial Derivatives. *Theor. Comput. Sci.* 22, pages 317–330, 1983.
- [27] P. Beame and T. Pitassi. Propositional Proof Complexity: Past, Present, and Future. In *Bulletin of the European Association for Theoretical Computer Science*, Vol. 65 (June), pages 66–89, 1998.
- [28] M. Bellare, O. Goldreich, and E. Petrank. Uniform Generation of NP-Witnesses Using an NP-Oracle. *Information and Computation*, Vol. 163, pages 510–526, 2000.
- [29] M. Bellare, O. Goldreich, and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27 (3), pages 804–915, 1998. Extended abstract in *36th FOCS*, 1995.
- [30] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the Theory of Average Case Complexity. *Journal of Computer and System Science*, Vol. 44 (2), pages 193–219, 1992.
- [31] A. Ben-Dor and S. Halevi. In *2nd Israel Symp. on Theory of Computing and Systems*, IEEE Computer Society Press, pages 108–117, 1993.
- [32] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything Provable Is Probable in Zero-Knowledge. In *Crypto88*, Springer-Verlag Lecture Notes in Computer Science (Vol. 403), pages 37–56, 1990.
- [33] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th ACM Symposium on the Theory of Computing*, pages 113–131, 1988.
- [34] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.
- [35] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006. Extended abstract in *36th STOC*, 2004.
- [36] E. Ben-Sasson and M. Sudan. Simple PCPs with Poly-log Rate and Query Complexity. In *37th ACM Symposium on the Theory of Computing*, pages 266–275, 2005.
- [37] L. Berman and J. Hartmanis. On Isomorphisms and Density of NP and Other Complete Sets. *SIAM Journal on Computing*, Vol. 6 (2), 1977, pages 305–322.
- [38] M. Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. *Journal of the ACM*, Vol. 14 (2), pages 290–305, 1967.
- [39] M. Blum and S. Kannan. Designing Programs That Check Their Work. In *21st ACM Symposium on the Theory of Computing*, pages 86–97, 1989.
- [40] M. Blum, M. Luby, and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47 (3), pages 549–595, 1993.
- [41] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.
- [42] A. Bogdanov, K. Obata, and L. Trevisan. A Lower Bound for Testing 3-Colorability in Bounded-Degree Graphs. In *43rd IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2002.

- [43] A. Bogdanov and L. Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM Journal on Computing*, Vol. 36 (4), pages 1119–1159, 2006. Extended abstract in *44th FOCS*, 2003.
- [44] A. Bogdanov and L. Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, Vol. 2 (1), 2006.
- [45] R. Boppana, J. Håstad, and S. Zachos. Does Co-NP Have Short Interactive Proofs? *Information Processing Letters*, Vol. 25 (May), pages 127–132, 1987.
- [46] R. Boppana and M. Sipser. The Complexity of Finite Functions. In *Handbook of Theoretical Computer Science: Volume A – Algorithms and Complexity*, J. van Leeuwen (ed.), MIT Press/Elsevier, pages 757–804, 1990.
- [47] A. Borodin. Computational Complexity and the Existence of Complexity Gaps. *Journal of the ACM*, Vol. 19 (1), pages 158–174, 1972.
- [48] A. Borodin. On Relating Time and Space to Size and Depth. *SIAM Journal on Computing*, Vol. 6 (4), pages 733–744, 1977.
- [49] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37 (2), pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.
- [50] L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Science*, Vol. 18, pages 143–154, 1979.
- [51] G. J. Chaitin. On the Length of Programs for Computing Finite Binary Sequences. *Journal of the ACM*, Vol. 13, pages 547–570, 1966.
- [52] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, Vol. 28, pages 114–133, 1981.
- [53] D. Chaum, C. Crépeau, and I. Damgård. Multi-party Unconditionally Secure Protocols. In *20th ACM Symposium on the Theory of Computing*, pages 11–19, 1988.
- [54] B. Chor and O. Goldreich. On the Power of Two-Point Based Sampling. *Jour. of Complexity*, Vol. 5, pages 96–106, 1989. Preliminary version dates 1985.
- [55] A. Church. An Unsolvable Problem of Elementary Number Theory. *Amer. J. of Math.*, Vol. 58, pages 345–363, 1936.
- [56] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. *SIAM Monographs on Discrete Mathematics and Applications*, 2001.
- [57] A. Cobham. The Intrinsic Computational Difficulty of Functions. In *Proc. 1964 International Congress for Logic Methodology and Philosophy of Science*, pages 24–30, 1964.
- [58] S. A. Cook. The Complexity of Theorem Proving Procedures. In *3rd ACM Symposium on the Theory of Computing*, pages 151–158, 1971.
- [59] S. A. Cook. An Overview of Computational Complexity. Turing Award Lecture. *CACM*, Vol. 26 (6), pages 401–408, 1983.
- [60] S. A. Cook. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control*, Vol. 64, pages 2–22, 1985.
- [61] S. A. Cook and R. A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *J. of Symbolic Logic*, Vol. 44 (1), pages 36–50, 1979.
- [62] D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *Journal of Symbolic Computation*, Vol. 9, pages 251–280, 1990.
- [63] T. M. Cover and G. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.
- [64] P. Crescenzi and V. Kann. A Compendium of NP Optimization problems. Available at <http://www.nada.kth.se/~viggo/wwwcompendium/>
- [65] R. A. DeMillo and R. J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, Vol. 7 (4), pages 193–195, 1978.
- [66] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22 (Nov.), pages 644–654, 1976.
- [67] I. Dinur. The PCP Theorem by Gap Amplification. In *38th ACM Symposium on the Theory of Computing*, pages 241–250, 2006.
- [68] I. Dinur and O. Reingold. Assignment-Testers: Towards a Combinatorial Proof of the PCP-Theorem. *SIAM Journal on Computing*, Vol. 36 (4), pages 975–1024, 2006. Extended abstract in *45th FOCS*, 2004.
- [69] I. Dinur and S. Safra. The Importance of Being Biased. In *34th ACM Symposium on the Theory of Computing*, pages 33–42, 2002.
- [70] J. Edmonds. Paths, Trees, and Flowers. *Canad. J. Math.*, Vol. 17, pages 449–467, 1965.
- [71] S. Even. *Graph Algorithms*. Computer Science Press, 1979.
- [72] S. Even, A. L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Information and Control*, Vol. 61, pages 159–173, 1984.

- [73] U. Feige, S. Goldwasser, L. Lovász, and S. Safra. On the Complexity of Approximating the Maximum Size of a Clique. Unpublished manuscript, 1990.
- [74] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is Almost NP-Complete. *Journal of the ACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.
- [75] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, Vol. 29 (1), pages 1–28, 1999. Preliminary version in *31st FOCS*, 1990.
- [76] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd ACM Symposium on the Theory of Computing*, pages 416–426, 1990.
- [77] E. Fischer. The Art of Uninformed Decisions: A Primer to Property Testing. *Bulletin of the European Association for Theoretical Computer Science*, Vol. 75, pages 97–126, 2001.
- [78] G. D. Forney. *Concatenated Codes*. MIT Press, 1966.
- [79] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-Space Lower Bounds for Satisfiability. *Journal of the ACM*, Vol. 52 (6), pages 835–865, 2005.
- [80] L. Fortnow, J. Rompel, and M. Sipser. On the Power of Multi-Prover Interactive Protocols. In *3rd IEEE Symp. on Structure in Complexity Theory*, pages 156–161, 1988. See errata in *5th IEEE Symp. on Structure in Complexity Theory*, pages 318–319, 1990.
- [81] S. Fortune. A Note on Sparse Complete Sets. *SIAM Journal on Computing*, Vol. 8, pages 431–433, 1979.
- [82] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: A Research Annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.
- [83] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, Vol. 17 (1), pages 13–27, 1984. Preliminary version in *22nd FOCS*, 1981.
- [84] O. Gabber and Z. Galil. Explicit Constructions of Linear Size Superconcentrators. *Journal of Computer and System Science*, Vol. 22, pages 407–420, 1981.
- [85] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [86] J. von zur Gathen. Algebraic Complexity Theory. *Ann. Rev. Comput. Sci.*, Vol. 3, pages 317–347, 1988.
- [87] O. Goldreich. *Foundation of Cryptography—Class Notes*. Computer Science Dept., Technion, Israel, Spring 1989. Superseded by [91, 92].
- [88] O. Goldreich. A Note on Computational Indistinguishability. *Information Processing Letters*, Vol. 34 (May), pages 277–281, 1990.
- [89] O. Goldreich. Notes on Levin’s Theory of Average-Case Complexity. *ECCC*, TR97-058, 1997.
- [90] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Algorithms and Combinatorics Series (Vol. 17), Springer, 1999.
- [91] O. Goldreich. *Foundation of Cryptography: Basic Tools*. Cambridge University Press, 2001.
- [92] O. Goldreich. *Foundation of Cryptography: Basic Applications*. Cambridge University Press, 2004.
- [93] O. Goldreich. Short Locally Testable Codes and Proofs (Survey). *ECCC*, TR05-014, 2005.
- [94] O. Goldreich. On Promise Problems (a survey in memory of Shimon Even [1935-2004]). *ECCC*, TR05-018, 2005.
- [95] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33 (4), pages 792–807, 1986.
- [96] O. Goldreich, S. Goldwasser, and A. Nussboim. On the Implementation of Huge Random Objects. In *44th IEEE Symposium on Foundations of Computer Science*, pages 68–79, 2003.
- [97] O. Goldreich, S. Goldwasser, and D. Ron. Property Testing and Its Connection to Learning and Approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.
- [98] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 25 (1), pages 169–192, 1996. Preliminary version in *17th ICALP*, 1990.
- [99] O. Goldreich and L.A. Levin. Hard-Core Predicates for Any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

- [100] O. Goldreich, S. Micali, and A. Wigderson. Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38 (3), pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.
- [101] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.
- [102] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *ECCC*, TR95-050, 1995.
- [103] O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, pages 302–343, 2002. Extended abstract in *29th STOC*, 1997.
- [104] O. Goldreich and D. Ron. A Sublinear Bipartite Tester for Bounded Degree Graphs. *Combinatorica*, Vol. 19 (3), pages 335–373, 1999. Extended abstract in *30th STOC*, 1998.
- [105] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning Polynomials with Queries: The Highly Noisy Case. *SIAM J. Discrete Math.*, Vol. 13 (4), pages 535–570, 2000.
- [106] O. Goldreich, S. Vadhan, and A. Wigderson. On Interactive Proofs with a Laconic Provers. *Computational Complexity*, Vol. 11, pages 1–53, 2002.
- [107] O. Goldreich and A. Wigderson. Computational Complexity. In *The Princeton Companion to Mathematics*, forthcoming.
- [108] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28 (2), pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.
- [109] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985. Earlier versions date to 1982.
- [110] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, Vol. 17, pages 281–308, 1988.
- [111] S. Goldwasser and M. Sipser. Private Coins Versus Public Coins in Interactive Proof Systems. *Advances in Computing Research: A Research Annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 73–90, 1989. Extended abstract in *18th STOC*, 1986.
- [112] S. W. Golomb. *Shift Register Sequences*. Holden-Day, 1967. (Aegean Park Press, revised edition, 1982.)
- [113] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh-Vardy Codes. In *22nd IEEE Conference on Computational Complexity*, pages 96–108, 2007.
- [114] J. Hartmanis and R. E. Stearns. On the Computational Complexity of Algorithms. *Transactions of the AMS*, Vol. 117, pages 285–306, 1965.
- [115] J. Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. *Advances in Computing Research: A Research Annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 143–170, 1989. Extended abstract in *18th STOC*, 1986.
- [116] J. Håstad. Clique Is Hard to Approximate Within $n^{1-\epsilon}$. *Acta Mathematica*, Vol. 182, pages 105–142, 1999. Preliminary versions in *28th STOC* (1996) and *37th FOCS* (1996).
- [117] J. Håstad. Getting Optimal In-Approximability Results. *Journal of the ACM*, Vol. 48, pages 798–859, 2001. Extended abstract in *29th STOC*, 1997.
- [118] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A Pseudorandom Generator from Any One-way Function. *SIAM Journal on Computing*, Vol. 28 (4), pages 1364–1396, 1999. Preliminary versions by Impagliazzo et al. in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).
- [119] J. Håstad and S. Khot. Query Efficient PCPs with Perfect Completeness. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 610–619, 2001.
- [120] A. Healy. Randomness-Efficient Sampling Within NC1. *Computational Complexity*, in press. Preliminary version in *10th RANDOM*, 2006.
- [121] A. Healy, S. Vadhan, and E. Viola. Using Nondeterminism to Amplify Hardness. *SIAM Journal on Computing*, Vol. 35 (4), pages 903–931, 2006.
- [122] D. Hochbaum (ed.). *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, 1996.
- [123] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [124] S. Hoory, N. Linial, and A. Wigderson. *Expander Graphs and Their Applications*. Bull. AMS, Vol. 43 (4), pages 439–561, 2006.

- [125] N. Immerman. Nondeterministic Space Is Closed Under Complementation. *SIAM Journal on Computing*, Vol. 17, pages 760–778, 1988.
- [126] R. Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *36th IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.
- [127] R. Impagliazzo and L. A. Levin. No Better Ways to Generate Hard NP Instances Than Picking Uniformly at Random. In *31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.
- [128] R. Impagliazzo and A. Wigderson. $P = BPP$ If E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *29th ACM Symposium on the Theory of Computing*, pages 220–229, 1997.
- [129] R. Impagliazzo and A. Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *Journal of Computer and System Science*, Vol. 63 (4), pages 672–688, 2001.
- [130] R. Impagliazzo and M. Yung. Direct Zero-Knowledge Computations. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), pages 40–51, 1987.
- [131] M. Jerrum, A. Sinclair, and E. Vigoda. A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with Non-Negative Entries. *Journal of the ACM*, Vol. 51 (4), pages 671–697, 2004.
- [132] M. Jerrum, L. Valiant, and V. V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science*, Vol. 43, pages 169–188, 1986.
- [133] B. Juba and M. Sudan. Towards Universal Semantic Communication. *ECCC*, TR07-084, 2007.
- [134] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, Vol. 13, pages 1–46, 2004. Preliminary version in *35th STOC*, 2003.
- [135] N. Kahale. Eigenvalues and Expansion of Regular Graphs. *Journal of the ACM*, Vol. 42 (5), pages 1091–1106, 1995.
- [136] R. Kannan, H. Venkateswaran, V. Vinay, and A. C. Yao. A Circuit-Based Proof of Toda's Theorem. *Information and Computation*, Vol. 104 (2), pages 271–276, 1993.
- [137] M. Karchmer and A. Wigderson. Monotone Circuits for Connectivity Require Super-logarithmic Depth. *SIAM J. Discrete Math.*, Vol. 3 (2), pages 255–265, 1990. Preliminary version in *20th STOC*, 1988.
- [138] R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (eds.), Plenum Press, pages 85–103, 1972.
- [139] R. M. Karp and R. J. Lipton. Some Connections Between Nonuniform and Uniform Complexity Classes. In *12th ACM Symposium on the Theory of Computing*, pages 302–309, 1980.
- [140] R. M. Karp and M. Luby. Monte-Carlo Algorithms for Enumeration and Reliability Problems. In *24th IEEE Symposium on Foundations of Computer Science*, pages 56–64, 1983.
- [141] R. M. Karp and V. Ramachandran. Parallel Algorithms for Shared-Memory Machines. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, J. van Leeuwen (ed.), MIT Press/Elsevier, pages 869–942, 1990.
- [142] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [143] S. Khot and O. Regev. Vertex Cover Might Be Hard to Approximate to Within $2 - \epsilon$. In *18th IEEE Conference on Computational Complexity*, pages 379–386, 2003.
- [144] V. M. Khrapchenko. A Method of Determining Lower Bounds for the Complexity of Pi-Schemes. In *Matematicheskie Zametki*, Vol. 10 (1), pages 83–92, 1971 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR*, Vol. 10 (1), pages 474–479, 1971.
- [145] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.
- [146] D. E. Knuth. *The Art of Computer Programming*, Vol. 2 (*Seminumerical Algorithms*). Addison-Wesley, 1969 (first edition) and 1981 (second edition).
- [147] A. Kolmogorov. Three Approaches to the Concept of “The Amount of Information.” *Probl. of Inform. Transm.*, Vol. 1 (1), 1965.
- [148] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1996.

- [149] R. E. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, Vol. 22, 1975, pages 155–171.
- [150] C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17, pages 215–217, 1983.
- [151] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.
- [152] L. A. Levin. Universal Search Problems. *Problemy Peredaci Informacii* 9, pages 115–116, 1973 (in Russian). English translation in *Problems of Information Transmission* 9, pages 265–266. A better English translation appears in [221].
- [153] L. A. Levin. Randomness Conservation Inequalities: Information and Independence in Mathematical Theories. *Information and Control*, Vol. 61, pages 15–37, 1984.
- [154] L. A. Levin. Average Case Complete Problems. *SIAM Journal on Computing*, Vol. 15, pages 285–286, 1986.
- [155] L. A. Levin. Fundamentals of Computing. *SIGACT News*, Education Forum, special 100th issue, Vol. 27 (3), pages 89–110, 1996.
- [156] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, August 1993.
- [157] R. J. Lipton. New Directions in Testing. *Distributed Computing and Cryptography*, J. Feigenbaum and M. Merritt (eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematics Society, Vol. 2, pages 191–202, 1991.
- [158] N. Livne. All Natural NPC Problems Have Average-Case Complete Versions. *ECCC*, TR06-122, 2006.
- [159] C.-J. Lu, O. Reingold, S. Vadhan, and A. Wigderson. Extractors: Optimal up to Constant Factors. In *35th ACM Symposium on the Theory of Computing*, pages 602–611, 2003.
- [160] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan Graphs. *Combinatorica*, Vol. 8, pages 261–277, 1988.
- [161] M. Luby and A. Wigderson. Pairwise Independence and Derandomization. *Foundations and Trends in Theoretical Computer Science*, Vol. (4), 2005. Preliminary version: TR-95-035, ICSI, Berkeley, 1995.
- [162] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, Vol. 39 (4), pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.
- [163] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1981.
- [164] G. A. Margulis. Explicit Construction of Concentrators. *Prob. Per. Infor.*, Vol. 9 (4), pages 71–80, 1973 (in Russian). English translation in *Problems of Infor. Trans.*, pages 325–332, 1975.
- [165] S. Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *35th FOCS*, 1994.
- [166] G. L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Science*, Vol. 13, pages 300–317, 1976.
- [167] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin Games Using Hitting Sets. *Computational Complexity*, Vol. 14 (3), pages 256–279, 2005. Preliminary version in *40th FOCS*, 1999.
- [168] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [169] M. Naor. Bit Commitment Using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.
- [170] J. Naor and M. Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM Journal on Computing*, Vol. 22, pages 838–856, 1993. Preliminary version in *22nd STOC*, 1990.
- [171] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Application. In *21st ACM Symposium on the Theory of Computing*, pages 33–43, 1989.
- [172] M. Nguyen, S. J. Ong, and S. Vadhan. Statistical Zero-Knowledge Arguments for NP from Any One-Way Function. In *47th IEEE Symposium on Foundations of Computer Science*, pages 3–14, 2006.
- [173] N. Nisan. Pseudorandom Bits for Constant Depth Circuits. *Combinatorica*, Vol. 11 (1), pages 63–70, 1991.
- [174] N. Nisan. Pseudorandom Generators for Space Bounded Computation. *Combinatorica*, Vol. 12 (4), pages 449–461, 1992. Preliminary version in *22nd STOC*, 1990.

- [175] N. Nisan. $RL \subseteq SC$. *Computational Complexity*, Vol. 4, pages 1-11, 1994. Preliminary version in *24th STOC*, 1992.
- [176] N. Nisan and A. Wigderson. Hardness vs Randomness. *Journal of Computer and System Science*, Vol. 49 (2), pages 149-167, 1994. Preliminary version in *29th FOCS*, 1988.
- [177] N. Nisan and D. Zuckerman. Randomness Is Linear in Space. *Journal of Computer and System Science*, Vol. 52 (1), pages 43-52, 1996. Preliminary version in *25th STOC*, 1993.
- [178] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
- [179] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. In *20th ACM Symposium on the Theory of Computing*, pages 229-234, 1988.
- [180] N. Pippenger and M. J. Fischer. Relations Among Complexity Measures. *Journal of the ACM*, Vol. 26 (2), pages 361-381, 1979.
- [181] E. Post. A Variant of a Recursively Unsolvable Problem. *Bull. AMS*, Vol. 52, pages 264-268, 1946.
- [182] M. O. Rabin. Digitalized Signatures. In *Foundations of Secure Computation* (R. A. DeMillo et al. eds.), Academic Press, 1977.
- [183] M. O. Rabin. Digitalized Signatures and Public Key Functions as Intractable as Factoring. MIT/LCS/TR-212, 1979.
- [184] M. O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, Vol. 12, pages 128-138, 1980.
- [185] R. Raz. A Parallel Repetition Theorem. *SIAM Journal on Computing*, Vol. 27 (3), pages 763-803, 1998. Extended abstract in *27th STOC*, 1995.
- [186] R. Raz and A. Wigderson. Monotone Circuits for Matching Require Linear Depth. *Journal of the ACM*, Vol. 39 (3), pages 736-744, 1992. Preliminary version in *22nd STOC*, 1990.
- [187] A. Razborov. Lower Bounds for the Monotone Complexity of some Boolean Functions. In *Doklady Akademii Nauk SSSR*, Vol. 281 (4), pages 798-801, 1985 (in Russian). English translation in *Soviet Math. Doklady*, Vol. 31, pages 354-357, 1985.
- [188] A. Razborov. Lower Bounds on the Size of Bounded-Depth Networks over a Complete Basis with Logical Addition. In *Matematicheskie Zametki*, Vol. 41 (4), pages 598-607, 1987 (in Russian). English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, Vol. 41 (4), pages 333-338, 1987.
- [189] A. R. Razborov and S. Rudich. Natural Proofs. *Journal of Computer and System Science*, Vol. 55 (1), pages 24-35, 1997. Preliminary version in *26th STOC*, 1994.
- [190] O. Reingold. Undirected ST-Connectivity in Log-Space. In *37th ACM Symposium on the Theory of Computing*, pages 376-385, 2005.
- [191] O. Reingold, S. Vadhan, and A. Wigderson. Entropy Waves, the Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors. *Annals of Mathematics*, Vol. 155 (1), pages 157-187, 2001. Preliminary version in *41st FOCS*, pages 3-13, 2000.
- [192] H. G. Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. *Trans. AMS*, Vol. 89, pages 25-59, 1953.
- [193] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21 (Feb.), pages 120-126, 1978.
- [194] D. Ron. Property Testing. In *Handbook on Randomization*, Volume 2, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim (eds.), pages 597-649, 2001.
- [195] R. Rubinfeld and M. Sudan. Robust Characterization of Polynomials with Applications to Program Testing. *SIAM Journal on Computing*, Vol. 25 (2), pages 252-271, 1996.
- [196] M. Saks and S. Zhou. $BPP_{SPACE}(S) \subseteq DSPACE(S^{3/2})$. *Journal of Computer and System Science*, Vol. 58 (2), pages 376-403, 1999. Preliminary version in *36th FOCS*, 1995.
- [197] W. J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Science*, Vol. 4 (2), pages 177-192, 1970.
- [198] A. Selman. On the Structure of NP. *Notices Amer. Math. Soc.*, Vol. 21 (6), page 310, 1974.
- [199] J. T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, Vol. 27 (4), pages 701-717, October 1980.
- [200] R. Shaltiel. Recent Developments in Explicit Constructions of Extractors. In *Current Trends in Theoretical Computer Science: The Challenge of the New Century*, Vol. 1: *Algorithms and Complexity*, G. Paun, G. Rozenberg, and A. Salomaa (eds.), World Scientific, 2004. Preliminary version in *Bulletin of the EATCS* 77, pages 67-95, 2002.
- [201] R. Shaltiel and C. Umans. Simple Extractors for All Min-Entropies and a New Pseudo-Random Generator. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 648-657, 2001.

- [202] A. Shamir. $IP = PSPACE$. *Journal of the ACM*, Vol. 39 (4), pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.
- [203] C. E. Shannon. A Symbolic Analysis of Relay and Switching Circuits. *Trans. American Institute of Electrical Engineers*, Vol. 57, pages 713–723, 1938.
- [204] C. E. Shannon. A Mathematical Theory of Communication. *Bell Sys. Tech. Jour.*, Vol. 27, pages 623–656, 1948.
- [205] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell Sys. Tech. Jour.*, Vol. 28, pages 656–715, 1949.
- [206] A. Shpilka. Lower Bounds for Matrix Product. *SIAM Journal on Computing*, pages 1185–1200, 2003.
- [207] M. Sipser. A Complexity Theoretic Approach to Randomness. In *15th ACM Symposium on the Theory of Computing*, pages 330–335, 1983.
- [208] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.
- [209] R. Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *19th ACM Symposium on the Theory of Computing*, pages 77–82, 1987.
- [210] R. J. Solomonoff. A Formal Theory of Inductive Inference. *Information and Control*, Vol. 7 (1), pages 1–22, 1964.
- [211] R. Solovay and V. Strassen. A Fast Monte-Carlo Test for Primality. *SIAM Journal on Computing*, Vol. 6, pages 84–85, 1977. Addendum in *SIAM Journal on Computing*, Vol. 7, page 118, 1978.
- [212] D. A. Spielman. *Advanced Complexity Theory*, Lectures 10 and 11. Notes (by D. Lewin and S. Vadhan), March 1997. Available from <http://www.cs.yale.edu/homes/spielman/AdvComplexity/1998/aslect10.ps> and [lect11.ps](http://www.cs.yale.edu/homes/spielman/AdvComplexity/1998/aslect11.ps).
- [213] L. J. Stockmeyer. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, Vol. 3, pages 1–22, 1977.
- [214] L. Stockmeyer. The Complexity of Approximate Counting. In *15th ACM Symposium on the Theory of Computing*, pages 118–126, 1983.
- [215] V. Strassen. Algebraic Complexity Theory. In *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*, J. van Leeuwen (ed.), MIT Press/Elsevier, pages 633–672, 1990.
- [216] M. Sudan. Decoding of Reed Solomon Codes Beyond the Error-Correction Bound. *Journal of Complexity*, Vol. 13 (1), pages 180–193, 1997.
- [217] M. Sudan. Algorithmic Introduction to Coding Theory. Lecture notes, available from <http://theory.csail.mit.edu/~madhu/FT01/>, 2001.
- [218] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom Generators Without the XOR Lemma. *Journal of Computer and System Science*, Vol. 62 (2), pages 236–266, 2001.
- [219] R. Szelepcsenyi. A Method of Forced Enumeration for Nondeterministic Automata. *Acta Informatica*, Vol. 26, pages 279–284, 1988.
- [220] S. Toda. PP Is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, Vol. 20 (5), pages 865–877, 1991.
- [221] B. A. Trakhtenbrot. A Survey of Russian Approaches to *Perebor* (Brute Force Search) Algorithms. *Annals of the History of Computing*, Vol. 6 (4), pages 384–398, 1984.
- [222] L. Trevisan. Extractors and Pseudorandom Generators. *Journal of the ACM*, Vol. 48 (4), pages 860–879, 2001. Preliminary version in *31st STOC*, 1999.
- [223] L. Trevisan. On Uniform Amplification of Hardness in NP . In *37th ACM Symposium on the Theory of Computing*, pages 31–38, 2005.
- [224] V. Trifonov. An $O(\log n \log \log n)$ Space Algorithm for Undirected st -Connectivity. In *37th ACM Symposium on the Theory of Computing*, pages 623–633, 2005.
- [225] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Mathematical Society*, Ser. 2, Vol. 42, pages 230–265, 1936. A Correction, *ibid.*, Vol. 43, pages 544–546.
- [226] C. Umans. Pseudo-Random Generators for All Hardness. *Journal of Computer and System Science*, Vol. 67 (2), pages 419–440, 2003.
- [227] S. Vadhan. A Study of Statistical Zero-Knowledge Proofs. Ph.D. thesis, MIT, 1999. Available from <http://www.eecs.harvard.edu/~salil/papers/phdthesis-abs.html>.
- [228] S. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM Journal on Computing*, Vol. 36 (4), pages 1160–1214, 2006. Extended abstract in *45th FOCS*, 2004.

- [229] S. Vadhan. *Lecture Notes for CS 225: Pseudorandomness*, Spring 2007. Available from <http://www.eecs.harvard.edu/~salil>.
- [230] L. G. Valiant. The Complexity of Computing the Permanent. *Theoretical Computer Science*, Vol. 8, pages 189–201, 1979.
- [231] L. G. Valiant. Completeness Classes in Algebra. In *11th ACM Symposium on the Theory of Computing*, pages 249–261, 1979.
- [232] L. G. Valiant. A Theory of the Learnable. *CACM*, Vol. 27 (11), pages 1134–1142, 1984.
- [233] L. G. Valiant and V. V. Vazirani. NP Is as Easy as Detecting Unique Solutions. *Theoretical Computer Science*, Vol. 47 (1), pages 85–93, 1986.
- [234] J. von Neumann. First Draft of a Report on the EDVAC, 1945. Contract No. W-670-ORD-492, Moore School of Electrical Engineering, Univ. of Pennsylvania. Reprinted (in part) in *Origins of Digital Computers: Selected Papers*, Springer-Verlag, pages 383–392, 1982.
- [235] J. von Neumann, Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100, pages 295–320, 1928.
- [236] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.
- [237] I. Wegener. *Branching Programs and Binary Decision Diagrams – Theory and Applications*. SIAM Monographs on Discrete Mathematics and Applications, 2000.
- [238] A. Wigderson. The Amazing Power of Pairwise Independence. In *26th ACM Symposium on the Theory of Computing*, pages 645–647, 1994.
- [239] A. C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.
- [240] A. C. Yao. Separating the Polynomial-Time Hierarchy by Oracles. In *26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.
- [241] A. C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.
- [242] S. Yekhanin. Towards 3-Query Locally Decodable Codes of Subexponential Length. In *39th ACM Symposium on the Theory of Computing*, pages 266–274, 2007.
- [243] R. E. Zippel. Probabilistic Algorithms for Sparse Polynomials. In the *Proceedings of EUROSAM '79: International Symposium on Symbolic and Algebraic Manipulation*, E. Ng (ed.). Lecture Notes in Computer Science (Vol. 72), pages 216–226, Springer, 1979.
- [244] D. Zuckerman. Linear-Degree Extractors and the Inapproximability of Max-Clique and Chromatic Number. In *38th ACM Symposium on the Theory of Computing*, pages 681–690, 2006.

后 记

别了，汉斯——但愿你还活着或是存在着！你的前景并不乐观。令你卷入其中的群魔乱舞的岁月还将持续不少年头，我们不敢担保你能幸免。说实话，我们真的不在意这个问题是否能得到解决。肉体和精神上的种种挑战使你变得不平凡，并让你在肉体消亡后精神永存。在过去那些庄严的时刻，你出于对死亡的恐惧和对肉体的反抗，曾在梦中萌发出爱情的种子。而今在这场世界性的死亡盛宴上，这个弥漫在雨夜天空之上的强烈欲火中，是否有一天仍会升起那样的爱呢？

托马斯曼，魔山，晴天霹雳

我们希望本书能成功地展现计算复杂性领域中概念、结论及公开问题的迷人魅力。我们也相信 21 世纪将见证该领域更多激动人心的发现，并激励读者做出更多贡献。在结束之前，还要谈一点想法。

正如 1.1.1 节所说，目前为止，复杂性理论主要是在基本计算现象之间建立联系，而非对基本问题给出明确答案。例如，NP-完全性理论与 P-vs-NP 问题之间，或伪随机性理论与单向函数存在性之间的关系（即使是在 $P \neq NP$ 的假设下）。普遍存在的挫败感来自于无法求解“绝对”型问题，而人们常对为什么会有这种失败感到好奇。

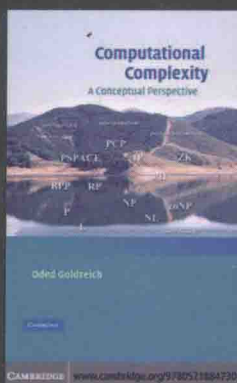
我们认为，这些失败大部分是由问题的困难性造成的。人们总是低估一些问题的难度，因为这些问题是如此的常见和引人关注。这里潜在的看法是如果一个问题常见且吸引人的，则求解该问题不应是困难的。这种观点十分可疑。我们认为，一个问题越是直观，就越难解决。因为直观的问题来自于生活中粗糙且混乱的现实，而非人为的、高度结构化的构造。事实上，自然的复杂性类和自然问题考虑的计算均来自于外部观察，从而缺乏任何内部结构。具体而言，复杂性类定义中使用的术语是算法的“外部行为”（即算法需要的资源），而非（问题的）“内部结构”。我们认为，复杂性理论定义中的“外部性质”使其难以解决。

关于“绝对”型（或“下界”）问题另一个困难是它们追求不可能的结论。也就是说，这些问题的定义中要求提供某种不存在的东西（即该类问题不存在高效的求解程序）。当然，证明某种事物不存在肯定比证明相关事物的存在性更困难（见 9.1 节）。然而，我们知道，在各个领域都可能要求证明某个事物不存在，特别是在复杂性理论中，但是，这种证明或者是平

凡的（见 4.1 节），或者可由一个复杂过程得到，该过程将原始问题转化为平凡问题。而后一种情况实际上是电路复杂性中许多成功结论的基础。因此，我们并非暗示复杂性理论中的“绝对”问题无法求解，而是建议对求解它们的困难性有一个直观认识。

一个确凿的事实是，根据近来的一些研究成果，困难问题是可以求解的，本书提到这样一些结论，并使我们改变之前的一些看法。这方面的例子包括 5.2.4 节中对数空间内遍历图的算法、9.3.2.3 节中 PCP 定理的另一种证明，以及定理 10.19 中对平均情况完全性的扩展。本书还提到参考文献[9, 113, 172, 242]中的一些结论，这极大地影响了我们的观点（虽然本书中表现得不十分明显）。

责任编辑：牛旭东 xdnui@ndip.cn
责任校对：苏向颖
封面设计：王 媛



Computational Complexity

A Conceptual Perspective

复杂性理论是计算机科学理论基础的核心。它主要研究计算任务的固有复杂性，即在有限的时间内（和/或其他有限的计算资源内）可以完成何种任务。

本书从概念的角度讨论复杂性理论。主要目的是使高年级本科生和研究生理解复杂性理论，或提供一本自学使用的教科书。本书还可供专业人士参考，因为其中阐述了复杂性理论的各种子领域，如困难放大、伪随机性以及概率证明系统。

作者在阐述各个子领域时，从该领域的直观问题着手，然后讨论这些问题的实际定义，为得到问题答案所使用的方法，以及答案中体现的思想。

Oded Goldreich是魏茨曼科学研究所的计算机教授，也是现任的Meyer W. Weisgal教授。他还是 *SIAM Journal on Computing*、*Journal of Cryptology* 以及 *Computation Complexity* 的编辑，出版了《现代密码学、概率证明与伪随机数》一书，以及两卷本的《密码学基础》。

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

此版本仅限在中华人民共和国境内
（不包括香港、澳门特别行政区及
台湾省）销售。



► 上架建议：密码学 计算理论 ◀

<http://www.ndip.cn>

ISBN 978-7-118-10387-8



9 787118 103878 >

定价：129.00 元